Journal Title
XX(X):1section*.1–28Learning to hit a pucktable.caption.47

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Composable energy policies for reactive motion generation and reinforcement learning

Julen Urain[1], Anqi Li[2], Puze Liu[1], Carlo D'Eramo[3,4] and Jan Peters[1]

Abstract

In this work, we introduce Composable Energy Policies (CEP), a novel framework for multi-objective motion generation. We frame the problem of composing multiple policy components from a probabilistic view. We consider a set of stochastic policies represented in arbitrary task spaces, where each policy represents a distribution of the actions to solve a particular task. Then, we aim to find the action in the configuration space that optimally satisfies all the policy components. The presented framework allows the fusion of motion generators from different sources: optimal control, data-driven policies, motion planning, handcrafted policies. Classically, the problem of multi-objective motion generation is solved by the composition of a set of deterministic policies, rather than stochastic policies. However, there are common situations where different policy components have conflicting behaviors, leading to oscillations or the robot getting stuck in an undesirable state. While our approach is not directly able to solve the conflicting policies problem, we claim that modeling each policy as a stochastic policy allows more expressive representations for each component in contrast with the classical reactive motion generation approaches. In some tasks, such as reaching a target in a cluttered environment, we show experimentally that CEP's additional expressivity allows us to model policies that reduce these conflicting behaviors.

A field that benefits from these reactive motion generators is the one of robot reinforcement learning. Integrating these policy architectures with reinforcement learning allows us to include a set of inductive biases in the learning problem. These inductive biases guide the reinforcement learning agent towards informative regions or improve collision safety while exploring. In our work, we show how to integrate our proposed reactive motion generator as a structured policy for reinforcement learning. Combining the reinforcement learning agent exploration with the prior-based CEP, we can improve the learning performance and explore safer.

Keywords

Reactive Motion Generation, Skill Composition, Reinforcement Learning, Energy Based Models, Multi-objective optimization

1 Introduction

Many robotic tasks deal with finding control actions satisfying multiple objectives. A seemingly simple task such as watering plants requires satisfying multiple objectives to perform it properly. The robot should reach the targets (the plants) with the watering can, avoid pouring water on the floor while approaching, and avoid colliding with and breaking the plant's branches by its arms. In contrast with sequential tasks (Sutton et al. 1999; Kaelbling and Lozano-Pérez 2011, 2013), in which the objectives to be satisfied are concatenated in time, in this work we consider tasks in which multiple geometric objectives must be satisfied in parallel.

The problem has been faced with a spectrum of solutions that balance between global optimality and computational complexity. Path planning methods (LaValle and Kuffner 2001; LaValle 2006; Kavraki et al. 1996) find a global trajectory from start to goal by a computationally intense Monte-Carlo sampling process. Trajectory optimization methods (Toussaint 2009; Ratliff et al. 2009; Kalakrishnan et al. 2011; Schulman et al. 2014; Mukadam et al. 2018) reduce the computational burden of planning methods by searching the global trajectory given initial trajectory candidates. These methods reshape the global trajectory to satisfy the objectives. However, they still require solving an optimization problem over long temporal horizon trajectories. The computational requirements of these algorithms limit the possibility of exploiting them for reactive motion generation. Computationally lighter, Model Predictive Control (MPC) methods (Morari and Lee 1999; Poignet and Gautier 2000; Williams et al. 2017; Bhardwaj et al. 2022) consider the problem of solving a short-horizon trajectory optimization problem reactively. Rather than assuming the problem of solving the trajectory optimization problem for the whole trajectory, these methods consider the problem of solving the trajectory optimization problem in a receding horizon reducing the computational requirements.

[1] Intelligent Autonomous Systems, TU Darmstadt, Germany
[2] Robot Learning Lab, University of Washington, USA
[3] Institute of Computer Science, University of Würzburg, Germany
[4] Hessian.AI, The Hessian Center for Artificial Intelligence, Germany

Corresponding author:
Julen Urain TU Darmstadt, FG IAS, Hochschulstr. 10, 64289 Darmstadt (Germany)
Email: julen@robot-learning.de

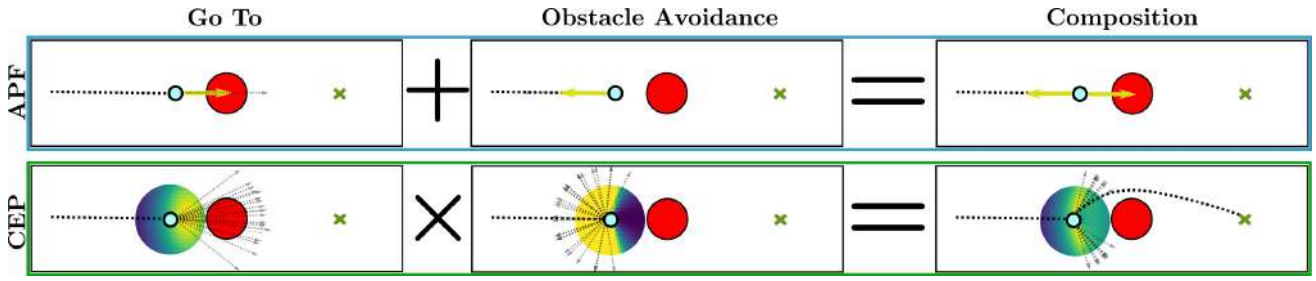


Figure 1. Visual Representation of modular control for Goto + Obstacle Avoidance. In the top box, we show Artificial Potential Fields (APF) (Khatib 1985). In the bottom box, we show Composable Energy Policies (CEP). In contrast, with the APF method, which sums deterministic actions (goto, avoid an obstacle), CEP computes the product of the policy distributions and then finds the maxima of the composition. The composition will provide a high probability to those actions that satisfy both components and low to the rest. *Robot: blue circle, obstacle: red circle, and target: green cross. Thick dotted line: performed trajectory, lightly dotted line: possible future trajectories.*[*]

Artificial Potential Fields (APF) methods (Khatib 1985; Ge and Cui 2002) and more recently Riemannian Motion Policies (RMP) methods (Ratliff et al. 2018; Kappler et al. 2018; Cheng et al. 2018; Bylard et al. 2021; Shaw et al. 2021; Aljalbout et al. 2021) are one of the most popular approaches for reactive motion generation in manipulators. In contrast with path planning or trajectory optimization methods, these methods propose to solve a myopic (one-step ahead) control problem. Given the problem is local, the computational cost is very low and they can be used with high control frequencies. These methods propose to solve the multi-objective control problem by the composition of a set of deterministic actions. Each action is computed to satisfy a particular objective. Then, the optimal composing action is found by solving a least-squares optimization problem given all the action components. The solution of the optimization problem can be analytically represented by a weighted sum of the action components. The sum of the actions defines a trade-off between the components with the weighting term (the metric) giving the relevance of each component. In these methods, it is common to have conflicting action components that make the robot get stuck.

Similarly to RMP and APF, we consider the problem of solving a one-step-ahead control problem. Considering a short-horizon problem reduces the variables to optimize and thus, we can guarantee sufficiently high control frequencies to be reactive for a high dimensional robot. In contrast with RMP and APF methods that assume deterministic policies to model the optimal behavior for each objective, we propose combining stochastic policies π_k. We hypothesize that considering arbitrary stochastic policies increases the expressivity on how to model the policies and we might represent policies that reduce the conflicting solutions in the composition. To find the composed action, we propose an optimization problem defined as a maximization over the log of the product of a set of stochastic policies (Paraschos et al. 2013; Haarnoja et al. 2018a)

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \log \left(\prod\nolimits_k \pi_k(\boldsymbol{a}|\boldsymbol{s}) \right), \qquad (1)$$

with action $\boldsymbol{a}$ and state $\boldsymbol{s}$. One can view the product of policies as a probabilistic instance of a logical conjunction (AND operator) (Du et al. 2020; Tasse et al. 2020) between the action distributions (see Figure 1 for visual representation). The product of policies will set a high probability to the actions that are likely to sample in all the components and low to the rest. If we would like to sample an action from the product of policies, we might apply Markov Chain Monte Carlo (MCMC) sampling process over the product of policies. In contrast, if we aim to obtain the action that satisfies best the composition, we can compute the maximum likelihood action (1).

Beyond local reactive navigation, policy composition has become a relevant approach to integrate inductive biases in robot Reinforcement Learning (RL) (Silver et al. 2018; Peng et al. 2019; Johannink et al. 2019; Li et al. 2021). These policy architectures allow the integration of reinforcement learning agents with prior knowledge. Rather than directly sampling the action commands for the robot, the reinforcement learning agent explores the parameter space of a structured policy. This structured model allows exploring safer or biasing the exploration towards informative regions. Nevertheless, prior methods assume explicit functions (Silver et al. 2018; Johannink et al. 2019; Li et al. 2021) to represent the structured policies. In contrast with previous methods, in our work, we propose to consider the optimization problem in (1) as the structured policy. We show empirically, that considering an implicit function to represent the structured policy allows us to explore with fewer collisions with respect to previous structured policies.

1.1 Contribution

A preliminary version of this paper was published as (Urain et al. 2021) in R:SS 2021. The current version complements the previous paper with multiple contributions:

1. We present an extended theoretical analysis relating RMP with Composable Energy Policies (CEP). Understanding how both methods are theoretically related allows a better comprehension of the obtained results and provides additional intuition for the practitioners to choose the method that fits their needs.
2. We provide a novel approach to model energy policies. In contrast with (Urain et al. 2021) that proposed handcrafted energy policies to deal with a set of

[*]Artificial Potential Fields can be framed as Composable Energy Policies, with each energy component represented by a quadratic function. For visualization purposes, we choose the classical representation of the sum of deterministic actions.

objectives, we derive the optimal control policies for a set of reward functions. Computing the policies as the solution of an optimization problem allows us to better understand what objectives each policy component aims to maximize.
3. We extend the experiments presented in (Urain et al. 2021), with additional evaluations. The additional experiments have been conducted to answer missing questions such as computational time of CEP, required optimization parameters, the performance of CEP under different RL algorithms or clarifying the benefits of the inductive biases in the reinforcement learning problem.
4. We have conducted new experiments in a real robot scenario. We evaluate the performance of CEP to solve a pick and place task with a real robot in a human-robot interaction environment. We measure the performance of the robot under human disturbances or online modifications of the picking and placing target positions.

Notation As our discussion will involve a set of policies and a set of spaces in which these policies are represented, we will use superscript (π^k) to represent the space in which the policy is and subscript (π_k) to represent the policy index. We represent the state by ($\boldsymbol{s}$) and the action by ($\boldsymbol{a}$). The space ($\mathcal{Q}$) represents the state-action space in the configuration $(\boldsymbol{s}^q, \boldsymbol{a}^q) \in \mathcal{Q}$ with $\boldsymbol{s}^q$ and $\boldsymbol{a}^q$ being, respectively, the state and action in the configuration space. ($\mathcal{X}_k$) represents the state-action space in the k task space $(\boldsymbol{s}^{x_k}, \boldsymbol{a}^{x_k}) \in \mathcal{X}_k$. $\boldsymbol{f}_q^x : \mathcal{Q} \to \mathcal{X}_k$ represents a transformation map from space $\mathcal{Q}$ to space $\mathcal{X}_k$. This map moves the state-action pairs from the configuration space to a task space.

1.2 Overview of APF and RMP

In reactive motion generation, we deal with the problem of generating the robot's motion online. The developed methods are required to have a low computational cost so that the robot responds fast to unexpected situations. Additionally, the generated motion should be able to deal with multiple tasks concurrently and represented in arbitrary spaces, such as avoiding collisions with multiple robot links, reaching a target with the end-effector, or avoiding joint limits. In APF and RMP, the optimal action is proposed to be computed by a weighted sum of the accelerations (RMP) or torques (APF) components solving each particular task. In RMP, each acceleration component is the output of a task space second-order dynamic system.

Let us assume a set of transformation maps $\boldsymbol{f}_q^{x_0}, \dots, \boldsymbol{f}_q^{x_K}$, mapping the position, velocity and acceleration in the configuration space $(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathcal{Q}$, to a set of task spaces $(\boldsymbol{x}_k, \dot{\boldsymbol{x}}_k, \ddot{\boldsymbol{x}}_k) \in \mathcal{X}_k$; with $\boldsymbol{f}_q^{x_k} : \mathcal{Q} \to \mathcal{X}_k$ represented by

$$
\begin{aligned}
\boldsymbol{x}_k &= \boldsymbol{\phi}_q^{x_k}(\boldsymbol{q}) \\
\dot{\boldsymbol{x}}_k &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\phi}_q^{x_k}(\boldsymbol{q}) = \boldsymbol{J}^{x_k}(\boldsymbol{q})\dot{\boldsymbol{q}} \\
\ddot{\boldsymbol{x}}_k &= \frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{\phi}_q^{x_k}(\boldsymbol{q}) = \boldsymbol{J}^{x_k}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}^{x_k}(\boldsymbol{q})\dot{\boldsymbol{q}} \approx \boldsymbol{J}^{x_k}(\boldsymbol{q})\ddot{\boldsymbol{q}}, \quad (2)
\end{aligned}
$$

with $\boldsymbol{J}^{x_k} = \partial\boldsymbol{\phi}_q^{x_k}(\boldsymbol{q})/\partial\boldsymbol{q}$ the Jacobian of the forward kinematic function $\boldsymbol{\phi}_q^{x_k}$. The transformation for the acceleration is usually approximated dropping out the curvature term $\dot{\boldsymbol{J}}^{x_k}(\boldsymbol{q})\dot{\boldsymbol{q}}$. Given the integration steps in control loops (running between 100 Hz and 1 kHz) are small it is a valid approximation (Ratliff et al. 2018).

Let us also consider a set of task space second-order dynamics systems $\ddot{\boldsymbol{x}}_k = \boldsymbol{g}^{x_k}(\boldsymbol{x}_k, \dot{\boldsymbol{x}}_k)$, with a metric $\boldsymbol{\Lambda}^{x_k}$ associated to them. APF and RMP methods deviates in how the metric $\boldsymbol{\Lambda}^{x_k}$ is represented and applied to weight the components. In APF, the metric is conditioned on the position, while in RMP, the metric is conditioned on both position and velocity. Given the kinematics model in (2), in (Ratliff et al. 2018), the dynamics and the metric in the configuration space $\mathcal{Q}$ are represented by

$$
\begin{aligned}
\boldsymbol{g}_k^q(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \boldsymbol{J}^{x_k\dagger}\boldsymbol{g}^{x_k}(\boldsymbol{\phi}_q^{x_k}(\boldsymbol{q}), \boldsymbol{J}^{x_k}(\boldsymbol{q})\dot{\boldsymbol{q}}) \\
\boldsymbol{\Lambda}_k^q &= \boldsymbol{J}^{x_k\intercal}\boldsymbol{\Lambda}^{x_k}\boldsymbol{J}^{x_k}, \quad (3)
\end{aligned}
$$

with $\boldsymbol{J}^{x_k\dagger}$ being the Jacobian pseudoinverse.

Finally, the acceleration in the configuration space is computed by a weighted-sum of all the dynamics systems represented in the configuration space

$$
\ddot{\boldsymbol{q}} = \left(\sum\nolimits_j \boldsymbol{\Lambda}_j^q\right)^{-1} \sum\nolimits_k \boldsymbol{\Lambda}_k^q \boldsymbol{g}_k^q. \quad (4)
$$

Instead, APF (Khatib 1985) methods do not compute the metric normalization

$$
\boldsymbol{\tau} = \sum\nolimits_k \boldsymbol{\Lambda}_k^q \boldsymbol{g}_k^q. \quad (5)
$$

The solution in (4) is proven to be the optimal action for a least-squares optimization problem (Ratliff et al. 2018)

$$
\ddot{\boldsymbol{q}}^* = \arg\min_{\ddot{\boldsymbol{q}}} \sum\nolimits_k \frac{1}{2} \left|\left|\ddot{\boldsymbol{q}} - \boldsymbol{g}_k^q\right|\right|_{\boldsymbol{\Lambda}_k^q}^2. \quad (6)
$$

Each dynamic component represents the policy to satisfy a particular objective, while the metric weights the influence of each component in the composed action.

2 Composable Energy Policies

We will first motivate our approach and, subsequently, we introduce the different elements our policy architecture is composed of.

2.1 Motivation

Composable energy policies (CEP) aims to provide a novel framework for multi-objective reactive motion generation. Our proposed method should be able to compute high frequency (100Hz-1kHz) control actions to apply in the robot. The computed action should be able to jointly satisfy multiple objectives. Additionally, we aim to model each component by an arbitrary stochastic policy.

The key idea of our proposed model is that in contrast to APF and RMP methods, we rather consider arbitrary stochastic policies to model each component. This leads to an optimisation problem where the cost of each component is no longer a quadratic function as in (6). We expect that, given that we can model each component arbitrarily, our method will be able to more easily find an action that satisfies all the objectives if the policies are chosen correctly. We visualize this intuition in Fig. 1.

2.2 Problem statement

Let us consider a set of stochastic policies, $\pi_k^{x_k}(\boldsymbol{a}^{x_k}|\boldsymbol{s}^{x_k})$, where each policy represents the optimal distribution in the action space to satisfy a particular objective. Each policy is represented in an arbitrary state-action space $\mathcal{X}_k$. Let us also consider a set of transformation maps $\boldsymbol{f}_{\boldsymbol{q}}^{\boldsymbol{x}_k}$

$$\begin{aligned}\boldsymbol{s}^{x_k} &= \boldsymbol{f}_{\boldsymbol{q}s}^{\boldsymbol{x}_k}(\boldsymbol{s}^q)\\ \boldsymbol{a}^{x_k} &= \boldsymbol{f}_{\boldsymbol{q}a}^{\boldsymbol{x}_k}(\boldsymbol{s}^q, \boldsymbol{a}^q),\end{aligned} \tag{7}$$

that relates the configuration space $\mathcal{Q}$, with the task spaces $\mathcal{X}_k$, in which each policy is defined. We aim to find the action in the configuration space, $\boldsymbol{a}^q$ that better satisfies all the policies. We frame the multi-objective reactive motion generation problem as an optimization problem defined by the policies and the task maps

$$\begin{aligned}\boldsymbol{a}^{q*} = \arg\max_{\boldsymbol{a}^q} &\log\left(\prod\nolimits_k \pi^{x_k}(\boldsymbol{a}^{x_k}|\boldsymbol{s}^{x_k})\right)\\ \text{s.t.}\quad &(\boldsymbol{a}^{x_k}, \boldsymbol{s}^{x_k}) = \boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q)\quad \forall k,\end{aligned} \tag{8}$$

with $\boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q) \equiv (\boldsymbol{f}_{\boldsymbol{q}a}^{\boldsymbol{x}_k}(\boldsymbol{s}^q, \boldsymbol{a}^q), \boldsymbol{f}_{\boldsymbol{q}s}^{\boldsymbol{x}_k}(\boldsymbol{s}^q))$ and $\boldsymbol{s}^q$ the current state in the configuration space. We assume $\boldsymbol{s}^q$ is given. The optimization in (8) represents our proposed reactive motion generation. Thus, we aim to solve this optimization in low computational time to guarantee high frequency control commands.

2.3 Composable energy policies method

Let us assume a set of independent stochastic policies $\pi_1(\boldsymbol{a}|\boldsymbol{s}), \ldots, \pi_K(\boldsymbol{a}|\boldsymbol{s})$ modeled by a Gibbs distribution

$$\pi_i(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(E_i(\boldsymbol{a}, \boldsymbol{s}))}{Z_i(\boldsymbol{s})}, \tag{9}$$

where $E : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is an arbitrarily represented energy function and $Z(\boldsymbol{s}) = \int_{\boldsymbol{a}} \exp(E(\boldsymbol{a}, \boldsymbol{s}))\mathrm{d}\boldsymbol{a}$ is the normalization factor. The choice of Gibbs distribution is not arbitrary. Gibbs distribution allows representing an arbitrary distribution by a suitable definition of the energy function E (Gibbs 1902). Additionally, computing the product of experts

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \prod\nolimits_k \pi_k(\boldsymbol{a}|\boldsymbol{s}) \propto \exp\left(\sum\nolimits_k E_k(\boldsymbol{a}, \boldsymbol{s})\right), \tag{10}$$

will end up in a weighted sum over the individual energy components in the exponential. Having the energy components linearly related is computationally beneficial. Given a set of energy components, in practice, we can parallelize the computation of all the components by multi-processing increasing the control frequency. Even if modeling the policy as a product of experts might seem an arbitrary choice, we show in Section 3 that, given a set of energy policies $\pi_1, \ldots, \pi_K$, the product of experts represents the distribution of the optimal behavior to satisfy all the policy components (Jaynes 1957).

2.3.1 Energy trees Inspired by APF (Khatib 1987) and RMP (Ratliff et al. 2018), we propose to model the composition of energies in different task spaces. In the composition proposed in (10), each energy function is considered to be in the same state-action space. However, in most of the robotics scenarios, we might be interested in composing together energies defined in different task spaces. Reaching a target while avoiding the obstacles, composes skills defined in different task spaces. To solve the task, we might require to define an attractor policy in the end-effector space of the robot and additional obstacle avoidance policies in different cartesian points in the links of the robot.

Our architecture is composed of two main components. First, we have a set of policies $\pi^{x_k}(\boldsymbol{a}^{x_k}|\boldsymbol{s}^{x_k})$, defined in different state-action task spaces $(\boldsymbol{s}^{x_k}, \boldsymbol{a}^{x_k}) \in \mathcal{X}_k$. Second, we consider a set of deterministic mappings that transform the state-action pairs in the configuration space $(\boldsymbol{s}^q, \boldsymbol{a}^q) \in \mathcal{Q}$ to the state-action task spaces $\mathcal{X}_k$, $\boldsymbol{f}_q^{x_k} : \mathcal{Q} \to \mathcal{X}_k$.

Consider the optimisation problem from (8). For a single policy component. The problem is written as (TODO)

$$\begin{aligned}\boldsymbol{a}^{q^*} &= \arg\max_{\boldsymbol{a}^q} \log \pi^{x_k}(\boldsymbol{a}^{x_k}|\boldsymbol{s}^{x_k})\\ \text{s.t.}\quad \boldsymbol{a}^{x_k} &= \boldsymbol{f}_{q^a}^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q) &(11)\\ \boldsymbol{s}^{x_k} &= \boldsymbol{f}_{q^s}^{x_k}(\boldsymbol{s}^q). &(12)\end{aligned}$$

The unconstrained representation of the optimization problem in (11) is

$$\boldsymbol{a}^{q^*} = \arg\max_{\boldsymbol{a}^q} \log \pi^{x_k}(\boldsymbol{f}_{q^a}^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q)|\boldsymbol{f}_{q^s}^{x_k}(\boldsymbol{s}^q)). \tag{13}$$

Moreover, given we are considering Gibbs distributions to represent each policy component, we can represent the optimization problem by

$$\boldsymbol{a}^{q^*} = \arg\max_{\boldsymbol{a}^q} E^{x_k}(\boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q)) - \log Z^{x_k}(\boldsymbol{s}^{x_k}). \tag{14}$$

The objective function is represented in terms of the energy function E^{x_k} and the log of the normalization function Z^{x_k}. We can follow similar derivation for the multi-objective problem. The objective function $\mathcal{J}$ for the unconstrained problem of (8) is represented as

$$\begin{aligned}\mathcal{J}(\boldsymbol{a}^q) &= \log \prod\nolimits_k \pi^{x_k}(\boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q))\\ &= \sum\nolimits_k \log \pi^{x_k}(\boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q))\\ &= \sum\nolimits_k E^{x_k}(\boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q)) - \log Z^{x_k}(\boldsymbol{s}^{x_k}).\end{aligned} \tag{15}$$

Additionally, from (7), given the normalization term does not depend on the action $\boldsymbol{a}^q$, we can neglect it from our objective function and optimize over the sum of the energy functions. In our work, we propose to compute the control action for our robot by the maximization of (15), $\boldsymbol{a}^q = \arg\max_{\boldsymbol{a}^q} \mathcal{J}(\boldsymbol{a}^q)$. In the general case, this optimization function lacks an analytical solution and we will use stochastic optimization methods to optimize it (De Boer et al. 2005).

Framing the robot control in terms of an implicit function has several interesting properties in contrast with the explicit counterpart (Khatib 1987; Ratliff et al. 2018). The first is that we are not constrained in the policy function. In CEP we can assume an arbitrary stochastic policy to represent each component, whereas explicit models assume deterministic policies. As we show in Section 3, RMP components can be thought as normal distributions from CEP lenses. This policy model freedom provides the practitioner with a much wider

range of opportunities to design policies or learn them with arbitrary energy based models (Urain et al. 2020; Florence et al. 2022). An implicit representation has additional relevant properties with respect to APF and RMP methods. To compute the desired acceleration in the configuration space; explicit methods require to invert the transformation map $\ddot{\boldsymbol{x}} \approx \boldsymbol{J}\ddot{\boldsymbol{q}}$, to move the desired acceleration from the task space to the configuration space. If the robot's configuration is close to a singularity, a small velocity in task space will result in a big velocity in joint space. In contrast, in CEP, given we are considering the implicit representation, we do not require to invert the transformation map as the energy is directly evaluated in the task space.

2.4 Optimization of composable energy policies

In the following, we introduce the algorithm we use to solve our optimization problem. In our problem, the optimal action is obtained by a maximization over the logarithm of the product of a set of expert policies (8). From the derivation in (15), if each policy is defined by a Gibbs distribution, the optimization function is

$$\boldsymbol{a}^{q*} = \arg\max_{\boldsymbol{a}^q} \sum\nolimits_k E^{x_k}(\boldsymbol{f}_q^{x_k}(\boldsymbol{a}^q, \boldsymbol{s}^q)), \tag{16}$$

with E^{x_k} being a set of given energy functions defined in arbitrary task spaces $\mathcal{X}^k$ and $\boldsymbol{f}_q^{x_k}$ being the transformation map that transforms a state-action pair in the configuration space $(\boldsymbol{s}^q, \boldsymbol{a}^q) \in \mathcal{Q}$ to the different task spaces $\mathcal{X}^k$.

We aim to solve the optimization in (16) in high control frequencies (100Hz-1kHz) to run it as a reactive motion generator. Additionally, the energy function might be non-differentiable. We propose to solve the optimization problem in (16) by cross-entropy methods (Botev et al. 2013). The proposed method is presented in Algorithm 1.

We initialize our algorithm transforming the state in the configuration space, $\boldsymbol{s}^q$ to the different task spaces $\boldsymbol{s}^{x_k}$. As shown in (7), the task space states do not depend on the action and we directly compute them given the configuration state. In terms of computational efficiency, it is relevant to compute the task space state out of the optimization loop as it might be computationally demanding (*usually, we compute the forward kinematics in this stage*). Then, we initialize the cross-entropy optimization for the configuration space action. We define a proposed sampling distribution $q(\boldsymbol{a}^q) = \mathcal{N}(\boldsymbol{a}^q|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\boldsymbol{I})$. Then, for I optimization steps, we first sample a set of N action candidates in the configuration space $\boldsymbol{a}_{0:N}^q$. To evaluate the samples, we first transform the samples to the set of K task spaces $\boldsymbol{a}_{0:K}^{x_k}$. In practise, we consider an affine map between $\boldsymbol{a}^q$ and $\boldsymbol{a}^{x_k}$.Thus, we can apply tensor multiplication and transform all the samples $\boldsymbol{a}_{0:N}^q$ to all the task spaces accelerations in a single step. Finally, the energies are computed on each energy component and the contributions summed.

In our problem, we consider two approaches to update the sampling distribution $q(\boldsymbol{a}^q)$. As proposed in (Botev et al. 2013), the mean and variance are updated by first selecting the M particles with the highest energy value. Then, the optimal mean and variance are computed by

$$\begin{aligned}\boldsymbol{\mu}^* &= \frac{1}{M}\sum_{m=0}^{M} \boldsymbol{a}_m^q \\ \boldsymbol{\sigma}^{*2} &= \frac{1}{M}\sum_{m=0}^{M} (\boldsymbol{a}_m^q - \boldsymbol{\mu}^*)^2.\end{aligned} \tag{17}$$

Alternatively, we also considered a soft update version. We represent the update by a reward weighted regression (Peters and Schaal 2007)

$$\begin{aligned}\boldsymbol{\mu}^* &= \frac{1}{\sum_{k=0}^{N}\omega_k}\sum_{n=0}^{N} \omega_n \boldsymbol{a}_n^q \\ \boldsymbol{\sigma}^{*2} &= \frac{1}{\sum_{k=0}^{N}\omega_k}\sum_{n=0}^{N} \omega_n (\boldsymbol{a}_n^q - \boldsymbol{\mu}^*)^2,\end{aligned} \tag{18}$$

with $\omega_n = \beta \exp(-\beta e_n^q)$. e_n^q is the total energy for the n action sample and $\beta > 0$ is a temperature parameter that scales the energies for the weighted mean and variance in (18).

Rather by cross-entropy (Botev et al. 2013) or by reward weighted regression (Peters and Schaal 2007), the mean and standard deviation for the next optimization step $\boldsymbol{\mu}_{i+1}$ is computed by smoothing the solution between the optimal one $\boldsymbol{\mu}^*$ and the previous one $\boldsymbol{\mu}_i$

$$\begin{aligned}\boldsymbol{\mu}_{i+1} &= \alpha\boldsymbol{\mu}_i + (1-\alpha)\boldsymbol{\mu}^* \\ \boldsymbol{\sigma}_{i+1} &= \alpha\boldsymbol{\sigma}_i + (1-\alpha)\boldsymbol{\sigma}^*.\end{aligned} \tag{19}$$

Algorithm 1: Composable Energy Policies

Given: N: Number of samples;
$\boldsymbol{s}^q$: Current state in configuration space;
K: Number of energy components;
$(\boldsymbol{f}_q^{x_1}, E^{x_1}), \ldots, (\boldsymbol{f}_q^{x_K}, E^{x_K}))$: Task maps and energies;
I: Optimization steps;
(μ_0, Σ_0): Initial sampling distribution mean and variance;
$(\boldsymbol{a}^*, e^*)$: Initial optimal action and energy;

for $k \leftarrow 1$ **to** K **do**
  $\boldsymbol{s}^{x_k} = \boldsymbol{f}_{qs}^{x_k}(\boldsymbol{s}^q)$; *Map configuration state to task states*
for $i \leftarrow 1$ **to** I **do**
  $\boldsymbol{a}_{0:N}^q \sim \mathcal{N}(\mu_i, \Sigma_i)$; *Sample N action candidates*
  for $k \leftarrow 1$ **to** K **do**
    $\boldsymbol{a}_n^{x_k} = \boldsymbol{f}_q^{x_k}(\boldsymbol{s}^q, \boldsymbol{a}_{0:N}^q)$; *Map actions to task spaces*
    $e_{0:N}^{x_k} = E^{x_k}(\boldsymbol{s}^{x_k}, \boldsymbol{a}_{0:N}^{x_k})$; *Evaluate energy*
  $e_{0:N}^q = \sum_{k=1}^{K} e_{0:N}^{x_k}$; *Sum all energies*
  $\mu_{i+1} \leftarrow \text{Update}_\mu(\mu_i, \boldsymbol{a}_{0:N}^q, e_{0:N}^q)$; *With* (17) *or* (18)
  $\Sigma_{i+1} \leftarrow \text{Update}_\Sigma(\Sigma_i, \boldsymbol{a}_{0:N}^q, e_{0:N}^q)$; *With* (17) *or* (18)
  $\boldsymbol{a}_i^*, e_i^* \leftarrow \arg_{\boldsymbol{a}^q}\max_e(e_{0:N}^q)$; *pick optimal action*
  if $e^* < e_i^*$ **then**
    $\boldsymbol{a}^* \leftarrow \boldsymbol{a}_i^*$;
    $e^* \leftarrow e_i^*$;
return $\boldsymbol{a}^*$;

Smoothing is often crucial to prevent premature shrinking of the sampling distribution (Botev et al. 2013).

It is common in model predictive control algorithms (Ohtsuka 2004) to assume that consecutive optimal control problems are similar to each other. This allows initializing the optimization with the previously computed optimal solution. In our work, we assume our optimization problem is myopic (*we only optimize for a single look ahead step*) and the energy functions might be non-continuous. Thus, we lack any guarantee of the consecutive optimal control problems to be similar to each other. In conclusion, we always initialize our optimization problem with zero mean and a sufficiently wide standard deviation.

3 An inference view on policy composition

In this section, we derive from an inference view the proposed optimization problem in (1). We additionally highlight the connections between RMP and CEP and prove that RMP methods can be considered a particular case of CEP.

Let us assume we aim to find the action distribution that satisfies in the optimal way a set of stochastic policies $\pi_k(\boldsymbol{a}|\boldsymbol{s})$. Similarly to control-as-inference (Rawlik et al. 2012; Levine 2018) approaches, we introduce an additional variable o_{π_k}. This variable is a binary random variable, where $o_{\pi_k} = 1$ denotes how likely state-action pair is optimal for the policy π_k and $o_{\pi_k} = 0$ denotes how unlikely. We choose to model the distribution over the "likelihood variable" o_{π_k} by

$$p(o_{\pi_k} = 1|\boldsymbol{s}, \boldsymbol{a}) \propto \pi_k(\boldsymbol{a}|\boldsymbol{s}) \propto \exp(E_k(\boldsymbol{s}, \boldsymbol{a})). \quad (20)$$

From (20), we can observe that for those cases in which the distribution is conditioned on the optimal state-action pairs for a particular policy π_k, the probability for $o_{\pi_k} = 1$ is going to be high. While if the action is not an action with a high likelihood for $\pi_k(\cdot|\boldsymbol{s})$, the optimality probability $p(o_{\pi_k} = 1|\boldsymbol{s}, \boldsymbol{a})$ will be low.

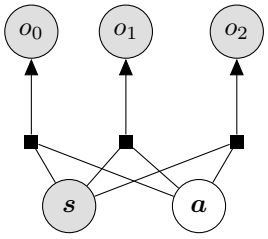


Figure 2. Graphical model for Composable Energy Policies. o_k is an auxiliary variable that represents the optimality of $\boldsymbol{s}_0$ and $\boldsymbol{a}_0$ for a particular policy.

Let us consider we aim to be optimal for a set of policies π_k. We can represent the Bayes net relating the optimality variables and the state and action as in Figure 2. The likelihood for the graphical model in Figure 2 can be computed as the product of the terms

$$p(\boldsymbol{s}, \boldsymbol{a}, o_{0:2}) = q(\boldsymbol{a})p(\boldsymbol{s}) \prod_{k=0}^{2} p(o_k|\boldsymbol{s}, \boldsymbol{a}). \quad (21)$$

with $p(\boldsymbol{s})$ and $q(\boldsymbol{a})$ the prior distributions for the state and the action consecutively.

Following the Bayes net, we can represent the posterior distribution over the action space when conditioned to $o_{\pi_k} = 1$ and $\boldsymbol{s} = \boldsymbol{s}_0$ by

$$p(\boldsymbol{a}|\boldsymbol{s} = \boldsymbol{s}_0, o_{0:K} = \mathbf{1}) \propto q(\boldsymbol{a}) \prod_{k=0}^{K} p(o_{\pi_k} = 1|\boldsymbol{s}_0, \boldsymbol{a})$$
$$\propto q(\boldsymbol{a}) \prod_{k=0}^{K} \pi_k(\boldsymbol{a}|\boldsymbol{s}) \propto q(\boldsymbol{a}) \exp\left(\sum_{k=0}^{K} E_k(\boldsymbol{a}, \boldsymbol{s})\right). \quad (22)$$

Considering the prior distribution over the action space to be uniform, $q(\boldsymbol{a}) = 1/\mathcal{A}$, we observe that the posterior distribution given all the optimality variables are 1 is the product of policies. By taking the *maximum a posteriori* estimate, $\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} p(\boldsymbol{a}|\boldsymbol{s} = \boldsymbol{s}_0, o_{0:K} = \mathbf{1})$, we compute the action that optimizes over the composition of all optimal distributions. The optimization problem is the one of (1).

In our work, we additionally consider a hard constraint that relates the state-action pairs in the configuration space with the state-action pairs in the task space (8). We integrate the constraints between the configuration space and the task space by a deterministic node in the graphical model (Figure 3). The deterministic node in the transformation will induce a delta distribution relating the state action pairs in the configuration space and in the task spaces

$$p(\boldsymbol{s}^{x_k}, \boldsymbol{a}^{x_k}|\boldsymbol{s}^q, \boldsymbol{a}^q) = \delta((\boldsymbol{s}^{x_k}, \boldsymbol{a}^{x_k}) - \boldsymbol{f}_q^{x_k}(\boldsymbol{s}^q, \boldsymbol{a}^q)). \quad (23)$$

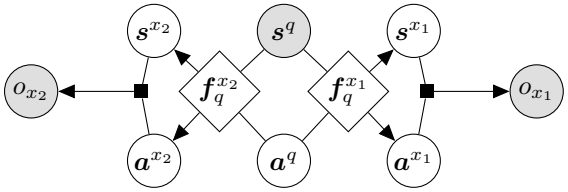


Figure 3. Graphical model for Composable Energy Policies with task space policies. o_{x_k} is an auxiliary variable that represents the optimality of $\boldsymbol{s}^{x_k}$ and $\boldsymbol{a}^{x_k}$ for a particular policy in that task space.

The likelihood function for the graphical model is represented by the product of the terms

$$p(\boldsymbol{s}^q, \boldsymbol{a}^q, \boldsymbol{s}^{x_{0:2}}, \boldsymbol{a}^{x_{0:2}}, o_{x_{0:2}}) =$$
$$p(\boldsymbol{s}^q)q(\boldsymbol{a}^q) \prod_{k=0}^{2} p(\boldsymbol{s}^{x_k}, \boldsymbol{a}^{x_k}|\boldsymbol{s}^q, \boldsymbol{a}^q)p(o_{x_k}|\boldsymbol{s}^{x_k}, \boldsymbol{a}^{x_k}). \quad (24)$$

Given that we aim to compute the posterior for $\boldsymbol{a}^q$ and assuming $\boldsymbol{s}^q$ and $o^{x_{0:2}}$ are given, we marginalize the joint distribution with respect to the rest of the variables

$$p(\boldsymbol{s}^q, \boldsymbol{a}^q, o_{x_{0:2}}) =$$
$$\int_{\boldsymbol{s}^{x_{0:2}}} \int_{\boldsymbol{a}^{x_{0:2}}} p(\boldsymbol{s}^q, \boldsymbol{a}^q, \boldsymbol{s}^{x_{0:2}}, \boldsymbol{a}^{x_{0:2}}, o_{x_{0:2}}) \mathrm{d}\boldsymbol{s}^{x_{0:2}} \mathrm{d}\boldsymbol{a}^{x_{0:2}}. \quad (25)$$

The relation between the configuration state-action pairs and the task space action pairs is given by (23). Given the relation

is defined by a delta distribution, the marginal distribution can be represented by a simple substitution of variables

$$p(\boldsymbol{s}^q, \boldsymbol{a}^q, o_{x_{0:2}}) = q(\boldsymbol{a}^q)p(\boldsymbol{s}^q)\prod_{k=0}^{2} p(o_{x_k}|\boldsymbol{s}^q, \boldsymbol{a}^q), \quad (26)$$

with $p(o_k|\boldsymbol{s}^q, \boldsymbol{a}^q) \propto \exp(E_k(\boldsymbol{f}_q^{x_k}(\boldsymbol{s}^q, \boldsymbol{a}^q)))$. Now, we can follow a similar derivation to (22) and compute the posterior distribution for the configuration space action $\boldsymbol{a}^q$.

3.1 Riemannian Motion Policies as Composable Energy Policies

The control-as-inference literature (Attias 2003; Toussaint 2009; Levine 2018) have widely studied the connections between the cost functions and the distributions related to them. From this viewpoint, RMP objective (6) can be framed as a particular case of (8) where each policy component is represented by a normal distribution. In the following, we derive the Riemannian motion policies from an inference perspective and show its relation with CEP.

Suppose each policy π is modelled by a normal distribution, where the mean, $\boldsymbol{g}^{x_k}$, is the desired optimal action in the task space $\mathcal{X}_k$, and the precision matrix, $\boldsymbol{\Lambda}^{x_k}$, is the metric on the task space

$$\pi(\boldsymbol{a}^{x_k}|\boldsymbol{s}^{x_k}) = \mathcal{N}(\boldsymbol{g}^{x_k}(\boldsymbol{s}^{x_k}), \boldsymbol{\Lambda}^{x_k}(\boldsymbol{s}^{x_k})). \quad (27)$$

We consider the action is defined by the acceleration, $\boldsymbol{a}^{x_k} = \ddot{\boldsymbol{x}}^k$ and the state by the position and velocity $\boldsymbol{s}^{x_k} = (\boldsymbol{x}^k, \dot{\boldsymbol{x}}^k)$.

In RMP methods the action in the task space $\mathcal{X}_k$ and in the configuration space $\mathcal{Q}$ are approximately related by the pseudo-inverse Jacobian of the forward kinematics, $\ddot{\boldsymbol{q}} \approx \boldsymbol{J}^{x_k^\dagger}\ddot{\boldsymbol{x}}^k$. Given the map is linear, the policy distribution in the task space (27) remains a normal distribution in the configuration space

$$\pi(\boldsymbol{a}^q|\boldsymbol{s}^q) = \mathcal{N}(\boldsymbol{J}^{x_k^\dagger}\boldsymbol{g}^{x_k}, \boldsymbol{J}^{x_k^\intercal}\boldsymbol{\Lambda}^{x_k}\boldsymbol{J}^{x_k}), \quad (28)$$

with the mean $\boldsymbol{g}^q = \boldsymbol{J}^{x_k^\dagger}\boldsymbol{g}^{x_k}$ and the precision matrix $\boldsymbol{\Lambda}^q = \boldsymbol{J}^{x_k^\intercal}\boldsymbol{\Lambda}^{x_k}\boldsymbol{J}^{x_k}$. In CEP, we assume the posterior distribution to maximize is modeled by the product of each policy (22). For the particular case in which every policy is represented by (28), the product of the policies remains a Gaussian, $p(\boldsymbol{a}|\boldsymbol{s} = \boldsymbol{s}_0, o_{0:K} = \mathbf{1}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ with

$$\begin{aligned}\boldsymbol{\mu} &= (\sum_j \boldsymbol{\Lambda}_j^q)^{-1}\sum_k \boldsymbol{\Lambda}_k^q\boldsymbol{g}_k^q \\ \boldsymbol{\Lambda} &= \sum_k \boldsymbol{\Lambda}_k^q.\end{aligned} \quad (29)$$

As we observe, the mean of the product of Gaussians is just a weighted-sum of the mean of each independent component and the precision matrix is the sum of each component. In CEP, the action is computed by a maximum a posteriori estimate over the posterior distribution $p(\boldsymbol{a}|\boldsymbol{s} = \boldsymbol{s}_0, o_{0:K} = \mathbf{1})$. For the particular case in which the posterior is the normal distribution in (29), the maximum a posteriori is the mean of the Gaussian

$$\ddot{\boldsymbol{q}}^* = (\sum_j \boldsymbol{\Lambda}_j^q)^{-1}\sum_k \boldsymbol{\Lambda}_k^q\boldsymbol{g}_k^q. \quad (30)$$

As expected, the solution is the one from the RMP (4). As shown in Urain et al. (2021), a similar derivation can be follow to represent APF as special cases of CEP.

In conclusion, we can derive the RMP solution as a special case of (8). To do so, we assume each policy component is represented by a normal distribution with the mean equal to the desired acceleration and the precision matrix equal to the metric. We hypothesize that in some tasks, representing all the policy components by a normal distribution might not be expressive enough to solve the task properly. Normal distributions assume that (1) there is a unique optimal action (the mean) to solve the task and (2) the quality of the actions is related to the Mahalanobis distance to the optima. While tasks like reaching a target might satisfy (1) and (2); tasks like obstacle avoidance might require richer representations to properly solve the task. Rather than limiting the policies to normal distributions, we consider arbitrary shape distributions to represent each objective. In the experimental section, we show empirically that for some tasks, modeling the policy with non-normal distributions might lead to better cooperation with the other components.

The scope of this paper is to study the relations and performances of one-step control horizon controllers. Nevertheless, given the clear relations of CEP with control-as-inference in longer horizon problems, we study these relations in the Appendix B.

4 A practical overview of energy policies

While the CEP framework can be used for arbitrary agents; in our work, we focus on the problem of generating motion for robot manipulators. The robot state in the configuration space $\boldsymbol{s}^q$ is represented by the robot's position $\boldsymbol{q}$, velocity $\dot{\boldsymbol{q}}$ and environment information $\boldsymbol{c}$ (obstacles position, obstacles shape, target pose). The action in the configuration space $\boldsymbol{a}^q$ is the robot's joint acceleration $\ddot{\boldsymbol{q}}$.

The energy policies are defined in a set of task spaces. We model the map from the configuration space to the different task spaces $\boldsymbol{f}_q^{x_k}$ by the robot's kinematics

$$\begin{aligned}\boldsymbol{x}_k &= \phi_q^{x_k}(\boldsymbol{q}) \\ \dot{\boldsymbol{x}}_k &= \boldsymbol{J}^k(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{x}}_k &= \boldsymbol{J}^k(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}^k(\boldsymbol{q})\dot{\boldsymbol{q}} \approx \boldsymbol{J}^k(\boldsymbol{q})\ddot{\boldsymbol{q}},\end{aligned} \quad (31)$$

with $\phi_q^{x_k}$ the forward kinematics to a given k task space and $\boldsymbol{J}^k(\boldsymbol{q}) = \partial\boldsymbol{x}_k/\partial\boldsymbol{q}$, the Jacobian for the given forward kinematics.

In CEP, we provide a framework to compose policies from different motion generation paradigms such as optimal control, imitation learning, movement primitives, reinforcement learning, or handcrafted policies. In this section, we introduce a practical overview of the different sources from which a policy component could be computed. In Section 4.1 we introduce a set of analytically computed energy policies to represent a set of basic local behaviors. Then, in Section 4.2, we introduce a set of possible methods to learn energy policies from data. Finally, in Section 4.3, we briefly introduced a set of methods to obtain energy policies from optimal control or reinforcement learning.

Task definition	Input space $(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}})$	transformation $(\boldsymbol{f})$	advantage function (A)	reward function (r)
reach target position	Cartesian Task Space $(\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3)$	$\boldsymbol{x} = \boldsymbol{x}$ $\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}$ $\ddot{\boldsymbol{x}} = \ddot{\boldsymbol{x}}$	$A(\ddot{\boldsymbol{x}}\|\boldsymbol{x}, \dot{\boldsymbol{x}}) = -\|\|\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_g(\boldsymbol{x}, \dot{\boldsymbol{x}})\|\|^2_{\boldsymbol{\Lambda}^{\ddot{x}}}$ $\ddot{\boldsymbol{x}}_g(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{2}{\Delta t^2}(\boldsymbol{x}_g - \boldsymbol{x} - \Delta t\dot{\boldsymbol{x}})$ $\boldsymbol{\Lambda}^{\ddot{x}} = \frac{\Delta t^4}{4}\boldsymbol{\Lambda}$	$r(\boldsymbol{x}) = -\|\|\boldsymbol{x} - \boldsymbol{x}_g\|\|^2_{\boldsymbol{\Lambda}}$
reach target orientation	Orientation Task Space $(SO(3), \mathbb{R}^3, \mathbb{R}^3)$	$\boldsymbol{\theta} = \text{LogMap}_{\boldsymbol{R}_g}(\boldsymbol{R})$ $\boldsymbol{\omega}' = \boldsymbol{R}_g^{-1}\boldsymbol{\omega}$ $\dot{\boldsymbol{\omega}}' = \boldsymbol{R}_g^{-1}\dot{\boldsymbol{\omega}}$	$A(\dot{\boldsymbol{\omega}}'\|\boldsymbol{\theta}, \boldsymbol{\omega}') = -\|\|\dot{\boldsymbol{\omega}}' - \dot{\boldsymbol{\omega}}_g(\boldsymbol{\theta}, \boldsymbol{\omega}')\|\|^2_{\boldsymbol{\Lambda}^{\dot{\omega}'}}$ $\dot{\boldsymbol{\omega}}_g(\boldsymbol{\theta}, \boldsymbol{\omega}') = \frac{2}{\Delta t^2}(\boldsymbol{\theta} - \Delta t\boldsymbol{\omega}')$ $\boldsymbol{\Lambda}^{\dot{\omega}'} = \frac{\Delta t^4}{4}\boldsymbol{\Lambda}$	$r(\boldsymbol{\theta}) = -\|\|\boldsymbol{\theta}\|\|^2_{\boldsymbol{\Lambda}}$
avoid obstacles	Cartesian Task Space $(\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3)$	$d_o = \|\|\boldsymbol{x} - \boldsymbol{x}_o\|\|$ $\hat{\boldsymbol{v}}_o = (\boldsymbol{x} - \boldsymbol{x}_o)/d_o$ $\dot{x}_p = \dot{\boldsymbol{x}} \cdot \hat{\boldsymbol{v}}_o$ $\ddot{x}_p = \ddot{\boldsymbol{x}} \cdot \hat{\boldsymbol{v}}_o$	$A(\ddot{x}_p\|\dot{x}_p, d_o) = \begin{cases} 0 & \text{if } \ddot{x}_p > \alpha^{\ddot{x}_p}(\dot{x}_p, d_o) \\ -\infty & \text{if } \ddot{x}_p \leq \alpha^{\ddot{x}_p}(\dot{x}_p, d_o) \end{cases}$ $\alpha^{\ddot{x}_p}(\dot{x}_p, d_o) = \frac{2}{\Delta t^2}(\alpha - d_o - \dot{x}_p\Delta t)$	$r(d_o) = \begin{cases} 0 & \text{if } d_o > \alpha \\ -\infty & \text{otherwise} \end{cases}$
avoid joint limits	Configuration Space $(\mathbb{R}^7, \mathbb{R}^7, \mathbb{R}^7)$	$\boldsymbol{q} = \boldsymbol{q}$ $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}$ $\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}$	$A(\ddot{\boldsymbol{q}}\|\dot{\boldsymbol{q}}, \boldsymbol{q}) = \begin{cases} 0 & \text{if } \ddot{\boldsymbol{q}} > \bar{\ddot{\boldsymbol{q}}} \text{ and } \ddot{\boldsymbol{q}} < \bar{\ddot{\boldsymbol{q}}} \\ -\infty & \text{otherwise} \end{cases}$ $\bar{\ddot{\boldsymbol{q}}}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \frac{2}{\Delta t^2}(\bar{\boldsymbol{q}} - \boldsymbol{q} - \dot{\boldsymbol{q}}\Delta t)$ $\bar{\ddot{\boldsymbol{q}}}(\dot{\boldsymbol{q}}, \boldsymbol{q}) = \frac{2}{\Delta t^2}(\bar{\boldsymbol{q}} - \boldsymbol{q} - \dot{\boldsymbol{q}}\Delta t).$	$r(\boldsymbol{q}) = \begin{cases} 0 & \text{if } \boldsymbol{q} > \boldsymbol{q} \text{ and } \boldsymbol{q} < \bar{\boldsymbol{q}} \\ -\infty & \text{otherwise} \end{cases}$
joint velocity control	Configuration Space $(\mathbb{R}^7, \mathbb{R}^7, \mathbb{R}^7)$	$\boldsymbol{q} = \boldsymbol{q}$ $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}$ $\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}$	$A(\ddot{\boldsymbol{q}}\|\boldsymbol{q}, \dot{\boldsymbol{q}}) = -\|\|\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_g(\dot{\boldsymbol{q}})\|\|^2_{\boldsymbol{\Lambda}^{\ddot{q}}}$ $\ddot{\boldsymbol{q}}_g(\dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}/\Delta t$ $\boldsymbol{\Lambda}^{\ddot{q}} = \Delta t^2\boldsymbol{\Lambda}$	$r(\dot{\boldsymbol{q}}) = -\|\|\dot{\boldsymbol{q}}\|\|^2_{\boldsymbol{\Lambda}}.$

Table 1. Resume of the proposed basic local reactive energies. To compute a particular energy, first the input position $\boldsymbol{x}$, velocity $\dot{\boldsymbol{x}}$ and acceleration $\ddot{\boldsymbol{x}}$ are transformed to a latent space by the maps in the third column. Then, the advantage is computed in the latent space. The advantage function represents the energy function of our policy. Last column shows the reward that each policy is trying to maximize.

4.1 Basic local reactive energies

In a previous work (Urain et al. 2021), we proposed handcrafted models to represent the local reactive energies. Handcrafted policies might lead to difficult parameter tuning when using them and lack an intuition of the objective they are trying to maximize. In this work, we propose to represent the energies as value functions maximizing a particular reward $r : \mathcal{S} \to \mathbb{R}$. In multiple problems, defining the behavior of the robot to solve a particular tasks with a reward function might be more intuitive and easier than defining it directly as a policy. If the task is easier to define in the state space rather than the action space, a reward function provides us with a natural form to describe the desired behavior. For example, in the case of collision avoidance, it is easier to represent the desired behavior in terms of the robot's position rather than with respect to the robot's acceleration. Modeling the energy of the policies in terms of the value function has been widely studied in the maximum entropy reinforcement learning community (Ziebart et al. 2010; Haarnoja et al. 2017). Under some assumptions, the value functions can be computed analytically. This approach allows the practitioner to understand the objective the policy is trying to maximize and provides additional intuition to tune the parameters.

The energies are represented by the optimal advantage function A for a one-step control horizon problem

$$A(\boldsymbol{a}|\boldsymbol{s}) = r(\boldsymbol{s}) + \mathbb{E}_{\rho(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}\left[V(\boldsymbol{s}')\right] - V(\boldsymbol{s}), \tag{32}$$

with $V(\boldsymbol{s}) = r(\boldsymbol{s})$ and $\rho(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ the transition dynamics. We can observe, that for the particular case of one-step control horizon with state dependant rewards, the advantage function is simply the expected value function given the transition dynamics.

Defining the state $\boldsymbol{s} = (\boldsymbol{x}, \dot{\boldsymbol{x}})$ by the position and the velocity and the action $\boldsymbol{a} = \ddot{\boldsymbol{x}}$ by the acceleration, we model the transition dynamics with the explicit Euler discretization of a linear dynamic system

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \dot{\boldsymbol{x}}_t\Delta t + \frac{1}{2}\Delta t^2\ddot{\boldsymbol{x}}_t \\ \dot{\boldsymbol{x}}_{t+1} &= \dot{\boldsymbol{x}}_t + \Delta t\ddot{\boldsymbol{x}}_t, \end{aligned} \tag{33}$$

with Δt being the step size. In the following we show that for some particular reward function, we can analytically derive the optimal advantage function that we exploit as the energy of our policy.

Target position Consider the problem of reaching a certain target position. The reward function to solve the problem can be modelled by the negative Mahalanobis distance to the target position $\boldsymbol{x}_g$

$$r(\boldsymbol{x}) = -||\boldsymbol{x} - \boldsymbol{x}_g||^2_{\boldsymbol{\Lambda}}. \tag{34}$$

The advantage function for the one-step ahead optimal control problem with state dependant reward is represented by

$$A(\boldsymbol{a}|\boldsymbol{s}) = \mathbb{E}_{\rho(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}[r(\boldsymbol{s}')]. \tag{35}$$

Given the dynamics in (33) are deterministic, the optimal advantage function can be computed analytically, by applying a change of variables in the reward (34)

$$A(\ddot{\boldsymbol{x}}|\boldsymbol{x}, \dot{\boldsymbol{x}}) = -||\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_g(\boldsymbol{x}, \dot{\boldsymbol{x}})||^2_{\boldsymbol{\Lambda}^{\ddot{x}}}, \tag{36}$$

with

$$\begin{aligned} \ddot{\boldsymbol{x}}_g(\boldsymbol{x}, \dot{\boldsymbol{x}}) &= \frac{2}{\Delta t^2}(\boldsymbol{x}_g - \boldsymbol{x} - \Delta t\dot{\boldsymbol{x}}) \\ \boldsymbol{\Lambda}^{\ddot{x}} &= \frac{\Delta t^4}{4}\boldsymbol{\Lambda}. \end{aligned} \tag{37}$$

The maximum acceleration of the advantage function is the one that moves a point in Δt to the $\boldsymbol{x}_g$. In our work we want to control the robot in fast control rates ($< 0.01s$). When the

robot is far from the target, reaching the target in that small Δt will require the robot to achieve very high accelerations. To avoid it, we model the new target position $\hat{\boldsymbol{x}}_g$ by

$$\hat{\boldsymbol{x}}_g = \boldsymbol{x} + \frac{\max(||\boldsymbol{x}_g - \boldsymbol{x}||, \alpha)}{||\boldsymbol{x}_g - \boldsymbol{x}||}(\boldsymbol{x}_g - \boldsymbol{x}). \tag{38}$$

The following equation projects the target position $\boldsymbol{x}_g$ to a ball centered in $\boldsymbol{x}$ and with a radius α. This way, the maximum Euclidean distance between the desired target and the current position is limited to α.

Target orientation We can apply a similar approach for reaching the desired orientation. In our work, we consider the orientation is represented in a Lie group $\boldsymbol{R} \in SO(3)$. Modeling a distance metric as the reward function is hard in the Lie Group given it is not a Euclidean space (Sola et al. 2018). To properly model the reward function, we first map the rotation to the Lie algebra $\mathfrak{so}(3)$ centered in the target orientation $\boldsymbol{R}_g \in SO(3)$ (*We transform the rotation matrix to the axis-angle representation*)

$$\boldsymbol{\theta} = \text{LogMap}_{\boldsymbol{R}_g}(\boldsymbol{R}), \tag{39}$$

with LogMap being the logarithmic map that moves a point in the Lie Group to the Lie algebra. The Lie algebra is an Euclidean space in which we can apply calculus. Given the Lie algebra is centered at the target $\boldsymbol{R}_g$, the desired target position in the Lie algebra is $\boldsymbol{\theta}_g = \boldsymbol{0}$; the origin. Thus, we can model the reward in $\mathfrak{so}(3)$ by

$$r(\boldsymbol{\theta}) = -||\boldsymbol{\theta}||^2_{\boldsymbol{\Lambda}}. \tag{40}$$

The reward function will maximize when $\boldsymbol{\theta} = \boldsymbol{0}$ and quadratically reduces with respect to the Euclidean distance in the Lie algebra.

To compute the optimal advantage function, we first transform the rotation velocity and accelerations from the world frame (*We compute the world frame velocity and acceleration in* (2)) to the target orientation frame $\boldsymbol{R}_g$

$$\begin{aligned} \boldsymbol{\omega}' &= \boldsymbol{R}_g^{-1}\boldsymbol{\omega} \\ \dot{\boldsymbol{\omega}}' &= \boldsymbol{R}_g^{-1}\dot{\boldsymbol{\omega}}. \end{aligned} \tag{41}$$

This map is known as the adjoint operation in the Lie Group theory (Sola et al. 2018). The linear dynamics in the Lie algebra are

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \boldsymbol{\omega}'_k\Delta t + \frac{1}{2}\Delta t^2\dot{\boldsymbol{\omega}}'_k \\ \boldsymbol{\omega}'_{k+1} &= \boldsymbol{\omega}'_k + \Delta t\dot{\boldsymbol{\omega}}'_k. \end{aligned} \tag{42}$$

Once everything is represented in the Lie algebra centered at $\boldsymbol{R}_g$, we can similarly to (36) compute the advantage function

$$A(\dot{\boldsymbol{\omega}}'|\boldsymbol{\theta}, \boldsymbol{\omega}') = -||\dot{\boldsymbol{\omega}}' - \dot{\boldsymbol{\omega}}_g(\boldsymbol{\theta}, \boldsymbol{\omega}')||^2_{\boldsymbol{\Lambda}^{\dot{\boldsymbol{\omega}}'}}, \tag{43}$$

with

$$\begin{aligned} \dot{\boldsymbol{\omega}}_g(\boldsymbol{\theta}, \boldsymbol{\omega}') &= \frac{2}{\Delta t^2}(\boldsymbol{\theta} - \Delta t\boldsymbol{\omega}') \\ \boldsymbol{\Lambda}^{\dot{\boldsymbol{\omega}}'} &= \frac{\Delta t^4}{4}\boldsymbol{\Lambda}. \end{aligned} \tag{44}$$

Similarly to (36), the acceleration maximizing the advantage is the one that sets the rotation to $\boldsymbol{R}_g$ in Δt. To bound the acceleration in the rotation we can also bound the target as in (38).

Obstacle avoidance We represent obstacle avoidance energy in the unidimensional space represented by the vector between a cartesian robot position $\boldsymbol{x}$ in a certain task space and the cartesian obstacle position $\boldsymbol{x}_o$. We compute this energy for every combination of a set of task space points P and a set of obstacles O. The total obstacle avoidance energy components are $P \times O$.

We first compute the distance to the obstacles and the vector pointing to the obstacle

$$\begin{aligned} d_o &= ||\boldsymbol{x} - \boldsymbol{x}_o|| \\ \hat{\boldsymbol{v}}_o &= (\boldsymbol{x} - \boldsymbol{x}_o)/d_o, \end{aligned} \tag{45}$$

with d_o being the distance and $\hat{\boldsymbol{v}}_o$ the vector pointing to the obstacle.

We define the obstacle avoidance reward function by

$$r(d_o) = \begin{cases} 0 & \text{if} \quad d_o > \alpha \\ -\infty & \text{if} \quad d_o \leq \alpha \end{cases}, \tag{46}$$

with α being a parameter that represents the minimum allowed distance to the obstacle. The proposed reward function allows the robot to be in any position except those that approximate to the obstacle to a distance below α.

To represent the advantage function, we first compute the velocity and acceleration projected in the vector $\hat{\boldsymbol{v}}_o$

$$\begin{aligned} \dot{x}_p &= \dot{\boldsymbol{x}} \cdot \hat{\boldsymbol{v}}_o \\ \ddot{x}_p &= \ddot{\boldsymbol{x}} \cdot \hat{\boldsymbol{v}}_o. \end{aligned} \tag{47}$$

Given the dynamics in (33), the dynamics in the projected space are

$$\begin{aligned} d_{ok+1} &= d_{ok} + \dot{x}_{pk}\Delta t + \frac{1}{2}\Delta t^2\ddot{x}_{pk} \\ \dot{x}_{pk+1} &= \dot{x}_{pk} + \Delta t\ddot{x}_{pk}. \end{aligned} \tag{48}$$

Then, we represent the advantage function in the unidimensional space represented by the vector between the current task space point and the obstacle

$$A(\ddot{x}_p|\dot{x}_p, d_o) = \begin{cases} 0 & \text{if} \quad \ddot{x}_p > \alpha^{\ddot{x}_p}(\dot{x}_p, d_o) \\ -\infty & \text{if} \quad \ddot{x}_p \leq \alpha^{\ddot{x}_p}(\dot{x}_p, d_o) \end{cases}, \tag{49}$$

with

$$\alpha^{\ddot{x}_p}(\dot{x}_p, d_o) = \frac{2}{\Delta t^2}(\alpha - d_o - \dot{x}_p\Delta t). \tag{50}$$

Using the advantage function in(49) as the energy of a policy represents a uniformly distributed policy

$$\pi(\ddot{x}_p|\dot{x}_p, d_o) = \mathcal{U}(\alpha^{\ddot{x}_p}(\dot{x}_p, d_o), \infty) \propto \exp(A(\ddot{x}_p|\dot{x}_p, d_o)). \tag{51}$$

Joint limits avoidance Similarly to the collision avoidance energy, we apply a binary reward to bound the joint limits. We define by $\underline{\boldsymbol{q}}$ and $\bar{\boldsymbol{q}}$ the minimum and maximum joints. We represent the reward by

$$r(\boldsymbol{q}) = \begin{cases} 0 & \text{if} \quad \boldsymbol{q} > \underline{\boldsymbol{q}} \text{ and } \boldsymbol{q} < \bar{\boldsymbol{q}} \\ -\infty & \text{otherwise} \end{cases}. \tag{52}$$

Given the reward in (52), the advantage function is

$$A(\ddot{\boldsymbol{q}}|\dot{\boldsymbol{q}},\boldsymbol{q}) = \begin{cases} 0 & \text{if} \quad \ddot{\boldsymbol{q}} > \underline{\ddot{\boldsymbol{q}}} \text{ and } \ddot{\boldsymbol{q}} < \bar{\ddot{\boldsymbol{q}}} \\ -\infty & \text{otherwise} \end{cases}, \tag{53}$$

with

$$\begin{aligned} \underline{\ddot{\boldsymbol{q}}}(\dot{\boldsymbol{q}},\boldsymbol{q}) &= \frac{2}{\Delta t^2}(\underline{\boldsymbol{q}} - \boldsymbol{q} - \dot{\boldsymbol{q}}\Delta t) \\ \bar{\ddot{\boldsymbol{q}}}(\dot{\boldsymbol{q}},\boldsymbol{q}) &= \frac{2}{\Delta t^2}(\bar{\boldsymbol{q}} - \boldsymbol{q} - \dot{\boldsymbol{q}}\Delta t). \end{aligned} \tag{54}$$

Joint velocity control Due to the myopic behavior of CEP, if the robot moves too fast, it might not be able to adapt fast enough to avoid collisions. Thus, we are interested in constraining the velocity the robot can achieve. We define a reward for the configuration space velocity

$$r(\dot{\boldsymbol{q}}) = -||\dot{\boldsymbol{q}}||^2_{\boldsymbol{\Lambda}}. \tag{55}$$

Given the reward in (55), the advantage function is

$$A(\ddot{\boldsymbol{q}}|\boldsymbol{q},\dot{\boldsymbol{q}}) = -||\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_g(\dot{\boldsymbol{q}})||^2_{\boldsymbol{\Lambda}^{\ddot{q}}}, \tag{56}$$

with

$$\begin{aligned} \ddot{\boldsymbol{q}}_g(\dot{\boldsymbol{q}}) &= \dot{\boldsymbol{q}}/\Delta t \\ \boldsymbol{\Lambda}^{\ddot{q}} &= \Delta t^2 \boldsymbol{\Lambda}. \end{aligned} \tag{57}$$

All these control energies are purely local. As we have shown, the proposed energies try to maximize a one-step-ahead control horizon reward. While they perform well for local navigation with obstacles, some tasks require a longer horizon look ahead to properly solve the task. In these situations, our myopic policies might fail. Nevertheless, these "smarter" policies could be obtained from data, given some expert demonstrations are provided or by applying long horizon optimal control or reinforcement learning and fitting a value function that solves a long horizon problem. Then, we could integrate these policies as an additional component of our CEP.

4.2 Learning energy policies from data

A common approach to learn policies from data is by behavioural cloning. Given a set of state-action pairs demonstrations $\mathcal{D} : \{\boldsymbol{s}_i, \boldsymbol{a}_i\}_{i=0:N}$, the policy is learned by a conditioned maximum likelihood estimation

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}} \left[\log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})\right]. \tag{58}$$

While the most common case assumes a conditioned Gaussian distribution as a policy model π, several works consider more expressive policy models. In (Urain et al. 2020), a normalizing flow is used to model the policy distribution, while in (Florence et al. 2022) an EBM is proposed to model the policy and trained by contrastive divergence (Hinton 2002).

Alternatively, we can build complex energy policies from simply learned distributions

Mixture-of-Expert energies A possible option to represent multi-modal policy distributions is to build a mixture of energies policies. Given a set of already given energies $E_0, \dots, E_k$, we can compute the mixture of energies

$$E_M(\boldsymbol{a},\boldsymbol{s}) = \log \sum_k w_k(\boldsymbol{s}) \exp(E_k(\boldsymbol{a},\boldsymbol{s})), \tag{59}$$

with w_k the weighting term. For the particular case in which the energy E is quadratic, the energy policy in (59) is the energy of a Gaussian mixture model.

Negated energy policy A more conservative approach to defining policies is by negative energies. Given a certain policy distribution, we might want our algorithm not to follow that policy without properly specifying what should be the desired path to follow. Given that the policy is modeled by energies of a Boltzman distribution $\pi(\boldsymbol{a}|\boldsymbol{s}) \propto \exp(E(\boldsymbol{a},\boldsymbol{s}))$, the negated policy is straightforwardly computed by negating the energy, $\pi_{\text{not}}(\boldsymbol{a}|\boldsymbol{s}) \propto \exp(-E(\boldsymbol{a},\boldsymbol{s}))$. This policy will inform about the action the robot should not do, rather than what to do.

4.3 Q-function in optimal control and reinforcement learning

The previously proposed advantage functions only solve a one-step-ahead control problem. Nevertheless, for several problems, we require to solve a longer horizon optimization. CEP enables integration of longer horizon value functions as energy components. We can integrate value functions learned by reinforcement learning (Haarnoja et al. 2018b) or optimal control (Lutter et al. 2021).Alternatively, we could compute the distribution for an optimal trajectory distribution by particles as in Stein Variational MPC (Lambert et al. 2021) and exploit this multi-modal distribution as a guiding policy. These learned models can be afterward integrated with additional energy policies to deal with specific parts that were not covered in the RL or optimal control problem.

5 Composable energy policies for robot reinforcement learning

In Reinforcement Learning, we deal with the problem of finding the policy π that maximizes the accumulated reward, $\mathcal{R}$

$$\max_{\pi} \mathbb{E}_{p_\pi(\boldsymbol{s},\boldsymbol{a})}[\mathcal{R}(\boldsymbol{s},\boldsymbol{a})]. \tag{60}$$

with $\rho_\pi(\boldsymbol{s},\boldsymbol{a})$ being the stationary state action distribution, given some transition dynamics, $p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})$ and initial state distribution $p(\boldsymbol{s}_0)$. Applying reinforcement learning in real robot environments usually consider high dimensional state-action spaces and sparse rewards. Thus, finding a good policy might require many iterations in the environment before a desirable policy is found.

A common approach to reduce the sample complexity is by integrating as many priors as possible in the problem. Properly chosen priors might accelerate the learning process, biasing the exploration towards meaningful states. Additionally, with the proper priors, we could increase the safety guarantees in the exploration process.

There are multiple ways to integrate priors in a reinforcement learning problem. A common option is to

do reward shaping. Adding additional reward signals to the problem, we can guide the learning process to informative states. Another common option is assuming a set of expert demonstrations are given, we can pretrain our policy to match the expert demonstrations. This approach is known as behavioral cloning. In our work, we explore the option of using structured policies.

A structured policy can be represented as follows:

$$\begin{aligned} \boldsymbol{\psi} &\sim \pi_{\text{RL}}(\boldsymbol{\psi}|\boldsymbol{s}) \\ \boldsymbol{a} &= \pi_{\text{struct}}(\boldsymbol{s};\boldsymbol{\psi}). \end{aligned} \tag{61}$$

A structured policy allows modifying the action space in which the RL agent learns the policy. Rather than directly sampling an action $\boldsymbol{a}$ from the RL agent; we sample a set of parameters $\boldsymbol{\psi}$. Then, these parameters are input in a low-level structured policy π_{struct} and the action is computed. Through the action space transformation, structured policies allow a faster and safer learning process.

There are several type of structured policies. In residual policy learning (Johannink et al. 2019; Silver et al. 2018), after sampling an action from the RL agent, an expert policy π_E action is summed to bias the exploration towards meaningful regions, $\pi_{\text{struct}}(\boldsymbol{s},\boldsymbol{\psi}) = \pi_E(\boldsymbol{s}) + \boldsymbol{\psi}$. The RL agent learns the residual actions around the expert policy. In (Dalal et al. 2021) the parameters $\boldsymbol{\psi}$ select a Dynamic Movement Primitives (DMP) and sets some parameters of the DMP, such as the target. Then, the DMP is executed for a certain period, before sampling new parameters from π_{RL}.

In our work, we propose to model the low-level structured policy by the maximization over a composition of policies

$$\pi_{\text{struct}}(\boldsymbol{s},\boldsymbol{\psi}) = \arg\max_{\boldsymbol{a}} \log\left(\prod_{k=0}^{K} \pi_k(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\psi})\right). \tag{62}$$

In contrast with previous works that define an explicit model to represent the structured policy, we propose to model the structured policy by a maximization over an implicit function. In our approach, we first sample a set of parameters from the RL agent. Then, these parameters condition some of the policies on the objective function. Finally, we solve the optimization problem in (62) to obtain the action to apply in the system. Considering an implicit function to represent the low-level policy has multiple benefits. An important one is related to safe exploration. Given we are solving a search problem, we can guarantee the robot is not choosing an action that would move the robot to a collision. We could also set some prior policies that encourage smooth behaviors and we could avoid high trembling while exploring. In conclusion, the robot explores the parameter space of an objective function. Then, given we have set some prior knowledge in this objective function, we can solve an optimization problem and apply the optimal action satisfying the objective.

There are multiple choices to parameterize the objective function. We show an example of how the energy policies and the RL action can be combined.

A simple option is to parameterize a reaching policy. We choose the target reaching policy proposed in Section 4. We can both parameterize the target position $\boldsymbol{x}_g$ or the metric $\boldsymbol{\Lambda}$

$$\begin{aligned} A(\ddot{\boldsymbol{x}}|\boldsymbol{x},\dot{\boldsymbol{x}}) &= -||\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_g(\boldsymbol{x},\dot{\boldsymbol{x}})||^2_{\boldsymbol{\Lambda}^{\ddot{x}}} \\ \ddot{\boldsymbol{x}}_g(\boldsymbol{x},\dot{\boldsymbol{x}}) &= \frac{2}{\Delta t^2}(\boldsymbol{\psi}_{\boldsymbol{x}_g} - \boldsymbol{x} - \Delta t\dot{\boldsymbol{x}}) \\ \boldsymbol{\Lambda}^{\ddot{x}} &= \frac{\Delta t^4}{4}\boldsymbol{\psi}_{\boldsymbol{\Lambda}}. \end{aligned}$$

Parameterizing $\boldsymbol{x}_g$ allows the reinforcement learning policy to set the desired target location given the current state $\boldsymbol{s}$, while parameterizing $\boldsymbol{\Lambda}$ allows the reinforcement learning agent to weight the influence of this component. It is important to remark, that for those cases in which $\boldsymbol{\Lambda}$ is big, the influence of this component in the composition increases and then, the influence of the reinforcement learning action. When $\boldsymbol{\Lambda}$ is small the contribution of this component decays and the inductive biases will define the movement.

Nevertheless, we remark, that we are not limited to reaching policies. We could parameterize any energy policy presented in Section 4, probabilistic motion primitives (Paraschos et al. 2013) or handcrafted policies.

In our experiments we combine parameterized policies with fixed prior policies

$$\pi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{\psi}) = \prod_k \pi_{k\text{prior}}(\boldsymbol{a}|\boldsymbol{s}) \prod_j \pi_j(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{\psi}). \tag{63}$$

Integrating a parameterized reaching target policy with a fixed obstacle avoidance policy, allows the robot to explore with the guarantee of exploring safely. We also consider combining attractor policies to a certain target and parameterized policies. Combining both, the reinforcement learning agent explores while the attractor policy guides the robot to informative regions. The performance is similar to a residual policy, in which the RL agent learns in the residuals of the guiding policy.

6 Experimental evaluation

The experimental evaluation is split into three parts. In the first part (Section 6.1), we evaluate qualitatively the performance of CEP in a 2D navigation environment. The experiment is performed to provide a visual intuition on how CEP represents its policy.

In the second part (Section 6.2), we investigate the performance of CEP for local reactive navigation in cluttered environments. The experiments are performed in a 7 dof Kuka-LWR robot. In a set of simulated environments (Section 6.2.1), we evaluate how the energy composition performs by observing the success rate and the collisions in a set of obstacle avoidance environments. Additionally, we perform ablation studies to find the optimal parameters and also to find the maximum control frequencies. Then, in a real robot environment (Section 6.2.2), we investigate the performance of CEP to solve a pick and place task in a cluttered environment. We measure the performance under human disturbances, picking position changes and placing position changes.

In the third part (Section 6.3), we investigate the benefit of integrating CEP as a structured policy for robot reinforcement learning. We first evaluate the performance of CEP as a structured policy while learning how to hit a puck and place it in a target position (Section 6.3.1). We

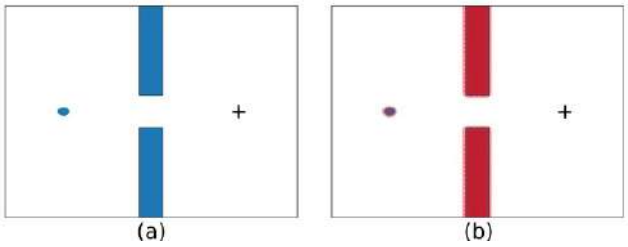


Figure 4. 2D navigation task. (a) Environment. The robot is represented by a blue circle, the walls by the rectangles and the target by the cross. (b) spherical obstacle bodies for the robot and the walls.

want to observe if using CEP as prior boosts the learning performance of an RL agent. Additionally, we want to evaluate if adding an obstacle avoidance prior reduces the number of collisions in the training. The experiments are performed for three MDP's that vary in the reward function.

6.1 Visual 2D particle environment

In the first experimental section, we aim to provide a visual understanding of the proposed policy composition. We investigate the composition of a set of energy policies proposed in Section 4 for a 2D navigation problem. We consider the toy environment in Figure 4 (a). We want to reach the target (cross) with the robot (blue circle) avoiding the walls (blue rectangles). To properly compute the obstacle avoidance, we represent the collision bodies for the walls and the robot with a set of spheres (Figure 4 (b)).

We model the composable energy policy with two component: a target reaching energy policy and a set of obstacle avoidance energy policies (one per each obstacle sphere in the wall). We model both energy policies with the proposed energy policies in Section 4. The attractor energy is a quadratic function. For obstacle avoidance, we consider N energy policies. Each energy function is a binary function that penalizes the actions that move the robot below a certain distance threshold with respect to the obstacles.

We visualize the probability density functions for each policy component and for the composition of them in Figure 5. Visualizing directly the distribution in the acceleration space is not informative. To provide an intuitive visualization, we plot the probability density function for the next state $p(\boldsymbol{s}')$, given the robot is sampling an action from a certain energy policy π

$$p(\boldsymbol{s}') = \mathbb{E}_{\pi(\boldsymbol{a}|\boldsymbol{s})}[\boldsymbol{f}(\boldsymbol{s}, \boldsymbol{a})], \tag{64}$$

with $\boldsymbol{f}$ being the linear dynamics in (33). Given these policies have been designed as the maximum entropy policies maximizing the next state reward, the next state distribution is naturally, $p(\boldsymbol{s}') \propto \exp(r(\boldsymbol{s}'))$.

We observe that the reaching target policy defines a normal distribution with the mean in an interpolation between the target position and the current position. The most likely action is the one that moves the robot to the mean position. The sharpness of the distribution is defined by the metric defined in Section 4. For the case of the obstacle avoidance policy, the next state distribution is a uniform distribution that sets the mass on a polytope defined by the shortest distances to the obstacles. This distribution will put zero mass to any action that moves the robot out of the polytope. For any action keeping the robot inside the polytope, the distribution remains constant.

The product of the two policies is a complex distribution that weights the influence of both. The obtained distribution is an attractor normal distribution truncated by the collision avoidance policy. We remark that rather than computing the maxima if we apply MCMC to sample from this distribution, the robot will never choose an action that collides against the obstacles and will sample actions that move towards the target with a higher probability.

6.2 Reaching through clutter environments

In the following experimental section, we investigate the performance of CEP for local navigation with a robot manipulator. The experimental section is divided between a simulated experimental section and a real robot experiment. The simulated experiments have been performed to answer the following questions:

Q1: Does energy policy composition increase the probability of reaching the target in a cluttered environment with respect to deterministic composition methods?

Q2: In CEP, the best action is found by stochastic optimization algorithms. How many particles do we need for a 7 dof robot? How many optimization steps?

In the real robot platform, we investigate the performance of CEP under disturbances. The main question we aim to answer is:

Q3: Is CEP able to reactively adapt to unmodelled disturbances, such as human physical interaction, changes in the placing position, or changes in the picking position?

6.2.1 Simulated reaching environments In the following, we present the simulated experiment to reach a certain target avoiding the obstacles. We perform the experiments with a 7 dof Kuka-LWR robot in 6 environments. The environments have been designed to be increasing in difficulty. The first environment has a single obstacle body, while the last one has 62 obstacle bodies. We visualize the considered environments in Figure 6. The robot is initialized in a randomized joint configuration and the motion is generated by a set of local motion generators, without additional global path planning algorithms. The episode ends when the robot reaches the target, collides against any obstacle or a certain time is pass. In our experiment, the robot is initialized in 100 randomly chosen initial configurations. We consider a control frequency of 250Hz for this experiment and the episode length is 30 seconds.

Policy setup For this experiment, we consider three possible multi-objective reactive motion generators: RMP (Ratliff et al. 2018), APF (Khatib 1985, 1987) and CEP. The three methods consider a set of policy components modeled in a set of task spaces. We compute the kinematics transformations in (2) with pinocchio (Carpentier et al. 2019). All the policies are local basic policies and there are no long-horizon planning components. The considered policy components are:

- A target reaching policy in the end-effector space;

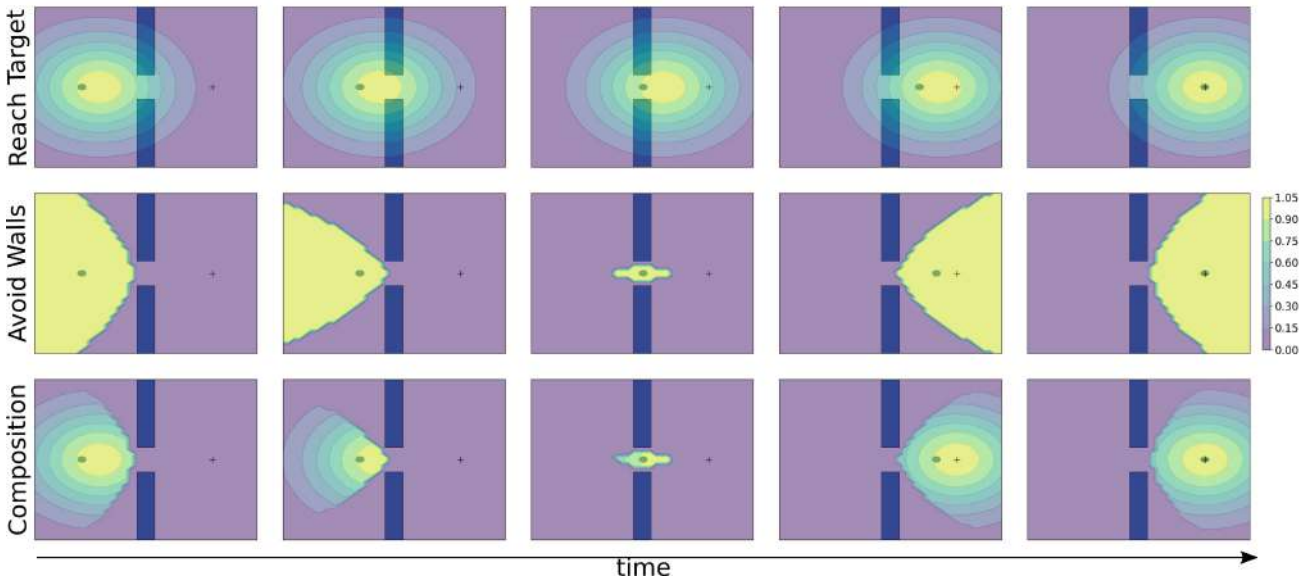


Figure 5. A visual representation of the next state distribution $p(\boldsymbol{s}')$ running a set of designed policies. We visualize the distribution for different states in a 2D navigation task. In the top: next state distribution after applying a reaching target policy. In the middle: next state distribution after applying an obstacle avoidance policy. In the bottom: next state distribution after applying the composition of a reaching target and an obstacle avoidance policy.

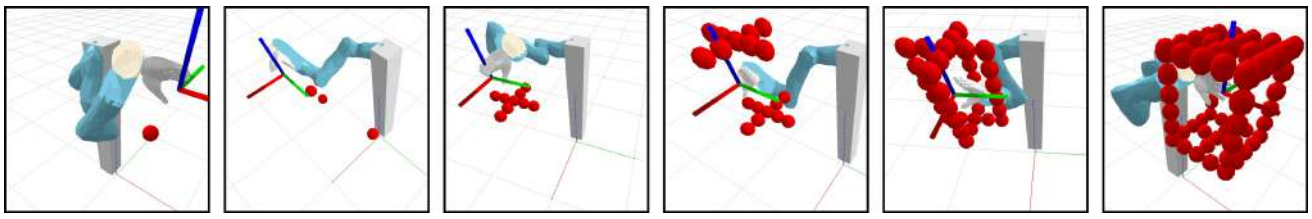


Figure 6. Simulated Environments for the reaching through clutter environments experiment. From left to right: 1 obstacle, 3 obstacles, Cross, Double Cross, Cage I and Cage II.

Methods	1 Obstacle		3 Obstacles		Cross		Double Cross		Cage I		Cage II	
	Success	Collide	Success	Collide	Success	Collide	Success	Collide	Success	Collide	Success	Collide
Riemannian Motion Policies (Ratliff et al. 2018)	100/100	0/100	99/100	0/100	93/100	0/100	87/100	0/100	29/100	0/100	5/100	0/100
Artificial Potential Fields (Khatib 1985, 1987)	100/100	0/100	98/100	0/100	91/100	0/100	46/100	0/100	2/100	0/100	0/100	0/100
Composable Energy Policies	**100/100**	**0/100**	**98/100**	**0/100**	**94/100**	**0/100**	**88/100**	**0/100**	**70/100**	**0/100**	**15/100**	**0/100**

Table 2. Results for 3D GoTo + Obstacle Avoidance Task. First three rows are the results from (Urain et al. 2021). We perform the same experiment with the robot hand included in row 4.

Methods	1 Obstacle		3 Obstacles		Cross		Double Cross		Cage I		Cage II	
	Success	Collide	Success	Collide	Success	Collide	Success	Collide	Success	Collide	Success	Collide
Riemannian Motion Policies (Ratliff et al. 2018)	100/100	0/100	89/100	0/100	71/100	0/100	63/100	0/100	11/100	0/100	0/100	0/100
Artificial Potential Fields (Khatib 1985, 1987)	100/100	0/100	90/100	0/100	68/100	0/100	21/100	0/100	0/100	0/100	0/100	0/100
Composable Energy Policies	**100/100**	**0/100**	**95/100**	**0/100**	**84/100**	**0/100**	**72/100**	**0/100**	**30/100**	**0/100**	**5/100**	**0/100**

Table 3. Results for 6D GoTo + Obstacle Avoidance Task. First three rows are the results from (Urain et al. 2021). We perform the same experiment with the robot hand included in row 4.

- $P \times O$ obstacle avoidance policies in a set of cartesian task spaces;
- Joint limits avoidance policy in the configuration space;
- Joint velocity limit policy in the configuration space.

We consider just position reaching policy for the 3D experiment and a full-pose reaching policy for the 6D experiment. For the obstacle avoidance policies, we model a collision body for the manipulator (Figure 7). The collision body is composed of 35 spheres with different radii. We remark that for the most complex environment, we have 62 collision spheres, thus, we have in total $35 \times 62 = 2170$ obstacle avoidance policies. Nevertheless, all of them are computed in batch using tensor multiplication.

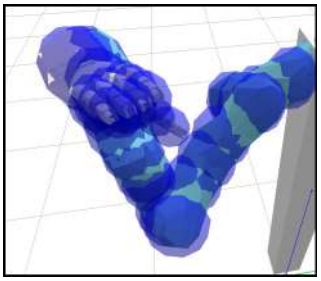


Figure 7. Collision body for the robot manipulator. The collision body is composed of 35 spheres with different radius.

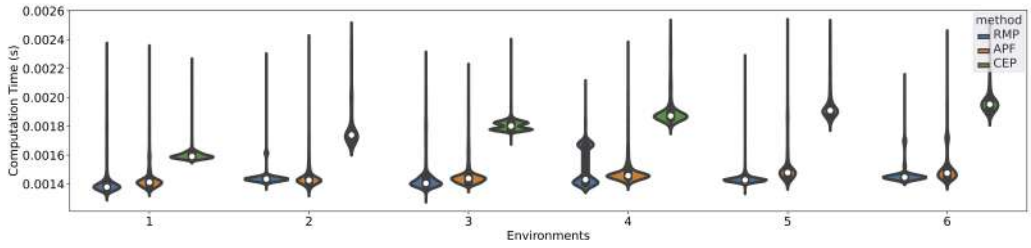


Figure 8. Controller's computation time for the six simulated environments in Figure 6. Measured for RMP, APF and CEP.

Comparative evaluation We initialize the experimental analysis evaluating the success rate and collisions in the 6 environments. We summarize the obtained results for the 3D go-to problem in table 2. We refer the reader to Figure 7 (Extension 1) for visualization of the experiments. The easy environments are easily solved by all the methods. We observe a success rate of almost 100% for the first three environments in all the cases. This result shows that simple scenarios can be easily solved with local reactive controllers and it is not required to solve a global trajectory planning problem. In complex scenarios, CEP performs better than the baselines. We can obtain a 70% success rate in the first cage environment and 15% in the second cage. We hypothesize that this could be related to how the obstacle avoidance policies are modeled. In the chosen baselines, the obstacle avoidance policies apply a repulsive force in a certain robot's link to avoid the obstacles. In highly cluttered environments, where the robot needs to move through narrow passages, these repulsive forces will push the robot far from the obstacles. Then, the robot is not able to get close to the narrow passage and it gets stuck in the entrance. In contrast, in our method, the obstacle avoidance policy defines a uniform distribution for the set of valid actions. This policy is more conservative. Rather than pushing the robot away from the obstacles, our policy penalizes, with very low probability, any action that moves a certain robot link close to the obstacles. Thus, in front of a narrow passage, the obstacle avoidance policy will only inform about those actions that are not valid but will not apply any repulsive force. We suggest that in the most complex environments, integrating the output of a trajectory optimization method could improve the robot's performance.

The performance worsens for the 6D go-to problem (Table 3). The orientation sets an important constraint in the possible final configurations and reduces the set of trajectories that solves the problem. CEP is able to perform relatively better than the baselines but it got less than 50% success rate in both cage environments, suggesting that in complex scenarios an additional global path planner should be integrated with CEP. It is important to remark, that both the deterministic baselines and our approach were able to properly impose the obstacle avoidance objective. We did not record any single collision in all the trials.

An important consideration for using CEP as reactive motion generators is its computational time. We compare the computation time for CEP with respect to RMP and APF. To properly compare them, we have considered the same amount of policies on the three cases and we used the same kinematics model. All the methods run in a *AMD Ryzen 9 3900* CPU. For CEP, we considered 50 particles and a single optimization step. We show the computational time in Figure 8. CEP is remarkably slower than RMP and APF. While previous methods consider an analytical solution for the optimal action, CEP requires solving an optimization problem. To do so, we require to evaluate a set of action particles and update a surrogate distribution (17). In our implementation, RMP and APF computes the solution in $0.0015s$ in average, while CEP computes the solution in around $0.002s$. Even if it is slower, we run CEP to 500Hz which is enough for reactive motion generation. Additionally, we remark that our implementations are not optimized and are running in an interpreted language. Thus, we expect faster computation times if the code is optimized and written in a compiled language. An additional point is related to the different computation times for the different environments. Our method requires an average computation time of $0.0016s$ for the first environment and $0.002s$ for the last one. The difference in computation is based on the number of collision spheres. In our work, collision avoidance is evaluated by computing the projected acceleration of a set of obstacle spheres in the robot with respect to a set of obstacles in the environment. The robot is composed of 35 obstacle spheres, while the environment varies from 1 to 62. To be computationally efficient, we apply tensor multiplication and compute the projected accelerations in parallel. Given N possible acceleration candidates in the configuration space, the number of projected accelerations are $N \times O \times 35$, where O is the number of obstacles in the environment. Given the number of collision spheres in the environment varies, the size of the tensor computing the projected accelerations also changes and thus, the required time for the tensor multiplication.

Conclusion 1 We have observed that the energy model flexibility provided by CEP improves the success rate with respect to previous methods. By choosing an appropriate energy policy, we can improve the cooperation between all the components and solve all the tasks jointly in a more successful way. Nevertheless, CEP requires solving an optimization problem by stochastic optimization methods, while previous methods consider an analytical solution. The stochastic optimization might reduce the computational efficiency of CEP and it will require more time to find a solution with respect to the analytic methods.

Ablation study CEP finds the optimal action by stochastic optimization. For I optimization steps, we sample N possible actions from the sampling distribution, we evaluate the objective function for each sample, and update the sampling distributions (Algorithm 1). The number of particles and the number of optimization steps will directly influence the performance of the robot. Also, the required computation time will vary depending on the number of samples and optimization steps. In the following experiment,

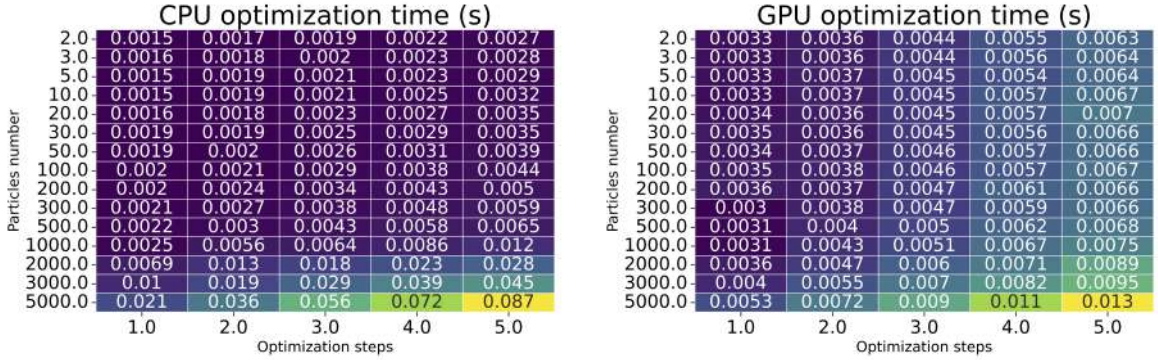


Table 4. CEP computation time for CPU and GPU. We consider the average computation time for 1-5 optimization steps.

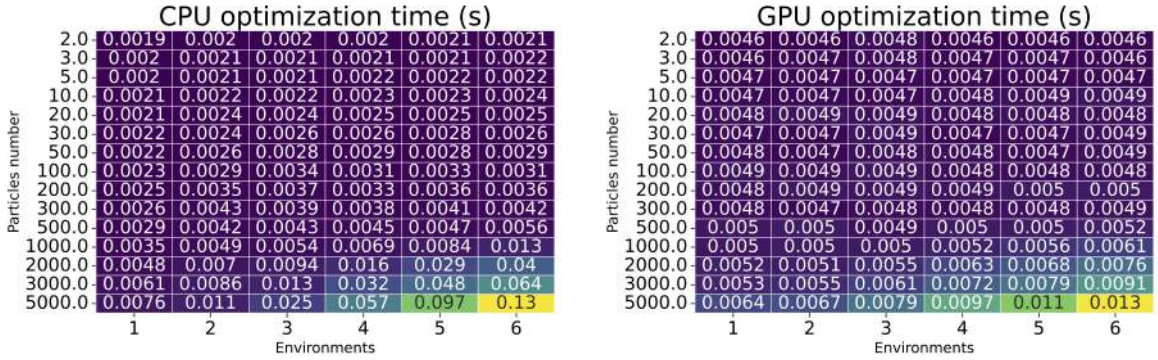


Table 5. CEP computation time for CPU and GPU. We show the variation of the mean computation time for the six environments

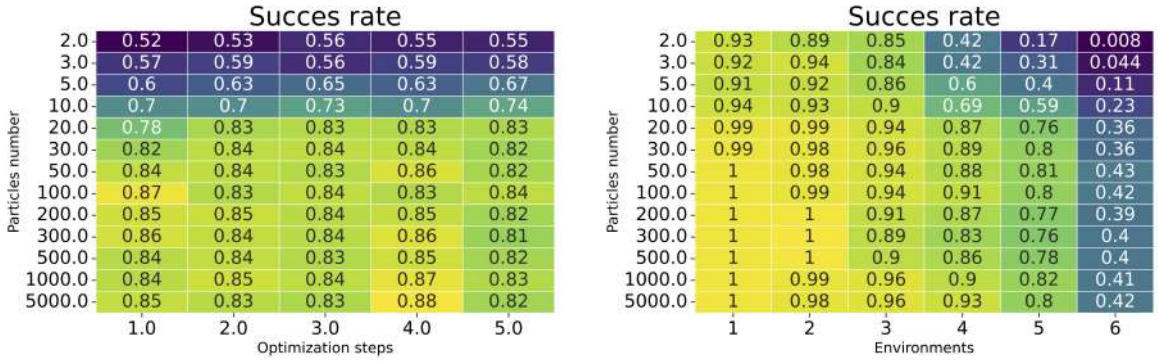


Table 6. CEP succes rate for reaching a target while avoiding obstacles. Left: Mean success rate for 1-5 optimization steps. Right: Mean success rate for the six environments.

we investigate the number of optimization steps and particles required for the reaching tasks represented in Figure 6. Even if the obtained solutions are specific for the chosen experiments, this ablation study is a relevant tool to estimate the required optimization parameters for similar experiments.

The experiments are performed in a *AMD Ryzen 3900* CPU and a *Nvidia GeForce RTX 2800* GPU. We consider both to investigate which hardware is more convenient for our problem. We present the obtained computational performance in table 4 and table 5. With few samples (< 500) the CPU is faster computationally than the GPU. Nevertheless, when the number of particles augment, the GPU outperformed the optimization time of the CPU. We can also observe that the required computation time linearly grows with the number of optimization steps in both CPU and GPU cases. From table. 5, we can observe that the computation effort remains constant for all the environments as long as the number of particles is small. Nevertheless, for a high amount of particles the computation frequency decays from the first environment to the sixth environment. For example, using a GPU with 5000 particles, we have an average computation time of 0.0064s (156Hz) for the first environment and 0.013s (77Hz) for the sixth environment. This change in the computation is directly connected with our obstacle avoidance energy. As previously introduced, we compute a tensor of $N \times O \times 35$ for the obstacle avoidance. For the simplest scenario, we have 1 obstacles sphere and for the most complex scenario, we have 62 spheres. For the case of $N = 5000$, in the simplest environment, we deal with a tensor of length $1.7e5$, while in the most complex environment, we require to compute a tensor of length $1.e7$.

In the tables 6, we introduce the results for an ablation study on the success rate. We investigate the required amount of particles and optimization steps to reach the targets without colliding. For the experiment, we execute CEP 100 times on each environment. We perform the same experiment for a different number of particles and a different number of optimization steps. The robot is initialized in a random configuration and reactively navigates to reach the target.

We show the mean success rate for all the environments in Table 6 (a) and the mean success rate for different number of optimization steps in Table 6 (b). For very few particles ($B = 2$) the robot performs poorly achieving a mean success rate of 0.54. Nevertheless, we can observe that it can solve properly the easiest environments and it has a success rate of 0.008 for the most complex one. We can also observe that for the cases in which few particles are used ($5 - 20$), the success rate improves if we use consider more optimization steps. After 50 particles, the success rate arrives in a plateau and an additional number of particles does not increase the success rate. We observe that while increasing optimization steps might have a direct impact with few particles (< 30) when considering a high amount of particles (> 30) a single optimization step is enough for solving the task at the maximum affordable success rate. We can also observe a clear pattern in the environments. While the simplest environments are solved with an almost $1.$ success rate, the performance decays up to a 0.4 success rate in the most complex environment. Due to the increase of obstacles, the robot gets stuck in local minima more often in complex environments with respect to simple ones. In these situations, a learned policy or a path planning algorithm could help the CEP, providing global guidance, while CEP solves the local reactive problem.

We remark that the variability in the success rate might be directly influenced by the stochasticity in the initial configuration. There are no predefined initial configurations, but rather the robot is initialized in arbitrary configurations.

Conclusion 2 From the tables 4 and tables 5, we conclude that for our experiments, if less than 500 particles are enough to solve the problem, we will choose the CPU while for more demanding tasks a GPU should be considered. From tables 6, we conclude that for the chosen experiments, 50 particles and one optimization step are enough to solve the tasks and we do not see further improvements when increasing the number of particles or the optimization steps. As we can observe, the computation times for 50 particles are sufficiently small to have control frequencies around $500Hz$. Nevertheless, we can observe how the computation requirements grow the more complex the environment is. We can estimate that for environments that are even more cluttered than ours, we might benefit from learning Signed Distance Functions (SDF) (Park et al. 2019) to reduce the computational requirements in the obstacle avoidance policy.

6.2.2 Pick-and-Place in cluttered environment In the following experiment, we evaluate the performance of CEP in a real robot scenario. We consider the problem of picking and place in a cluttered environment. To pick the object, the robot is required to navigate its hand through a narrow hole, grasp the object and navigate out of the narrow hole to leave the object in a plate. We remark there is no global path planning or trajectory optimization and the robot reactively computes the desired accelerations with a local controller. We present a visualization of the task in Figure 9. We have modeled the obstacle wall by 67 obstacle spheres (Figure 11). With this experiment, we aim to investigate the performance of CEP in complex environments as real-world human-robot interaction environments. We additionally evaluate the performance of RMP as baseline.

To control the robot, we use a CEP with a similar architecture to the simulated experiments:

- A target reaching policy in the end-effector space
- $P \times O$ obstacle avoidance policies in a set of cartesian task spaces
- Joint limits avoidance policy in the configuration space
- Joint velocity limit policy in the configuration space

The robot is controlled with 50 particles and a single optimization step.

To investigate the performance of CEP in the pick and place task, we measure the success rate and the execution time for picking and placing under 3 conditions. First, we assume there is a fixed target to pick and place and no human perturbing the robot. Second, the human applies physical perturbations to the robot and changes its position, and third, we track the human hand and dish to leave the object. Then, the picking and placing targets are changed online. We run the pick and place routine 30 times for every case. We present the results in Figure 10. The experiments can be visualized from Table 7 (Extension 2) to Table 7 (Extension 6).

On average, the robot is able to solve the pick and placing task under the three conditions with more than 75% success rate. We can observe that the robot is performing slightly better when no human perturbations or target modifications happened (see Figure 7 (Extension 2)). The failure cases are related to the robot getting stuck in an unrecoverable state (see Figure 7 (Extension 6)). For some initial configurations, the robot might enter wrongly in the not correct hole. Given CEP is myopic, it lacks any notion on how to escape from the hole and it gets stuck. When the human interacts with the robot it might move the robot to an unrecoverable state more often and then, the robot is not able to recover and gets stuck (see Figure 7 (Extension 3)). In the target modification case, we reactively change the picking position and the placing position (see Figure 7 (Extension 4-5)). During the picking, we find the robot used to get stuck if the picking point is too far from the holes. Once the robot moves its arm inside the hole, the maneuverability decays. This might result in the robot getting in some configurations in which it does not know how to get out of the hole.

On average, the performance is better for placing than for picking. In particular, when the placing target is modified, there is only the target reaching component having a big influence on the robot's motion. Nevertheless, when picking, the robot needs to trade-off between the obstacle avoidance component and the picking target reaching component. Due to this, the performance is better in the placing. We observe that CEP outperformed RMP in all situations. Similarly to simulated experiments, we consider that our proposed collision avoidance energy policy allows a better integration with the attractor policy in contrast with the repulsive collision avoidance policy from RMP. We consider that there are two easy fixes to improve the performance of the robot in the situations it gets stuck. First, we can easily combine longer horizon planning methods with CEP, to escape local minima. Additionally, we can learn specific energy policies that guide the robot through narrow passages. These energy policies will lead the robot's behavior when the robot is in a difficult passage, but won't influence the performance of the robot when is far from the narrow passages.

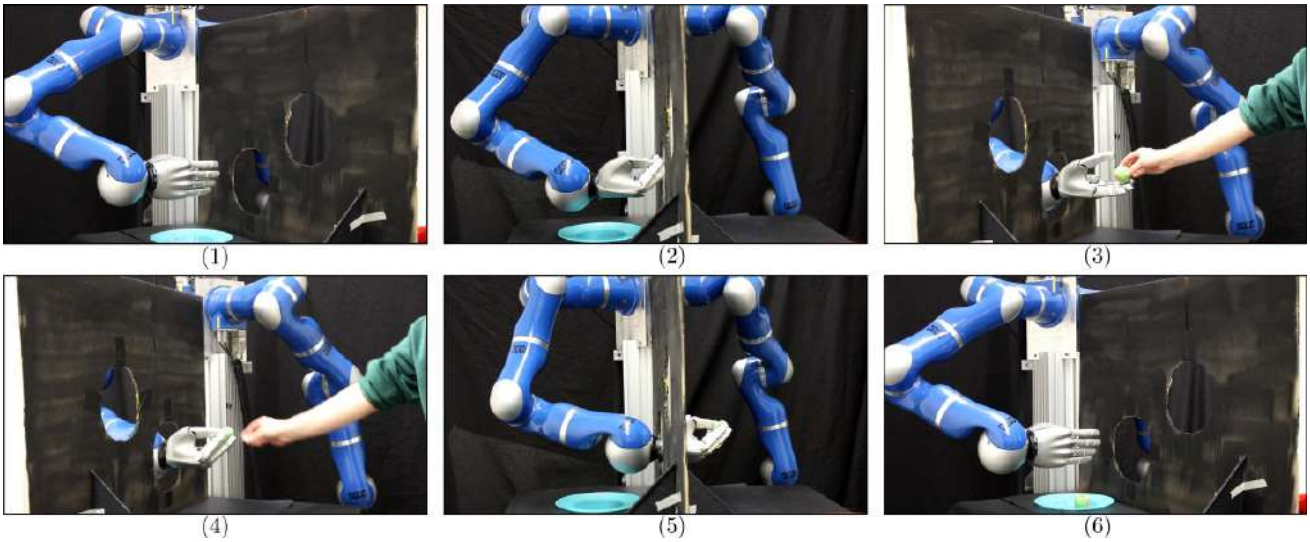


Figure 9. A visual representation of the pick and place task in a real robot environment. The robot is initialized in the left side. It should reach to the other side through the holes to pick the object. Then, move back to the left side to place the object.

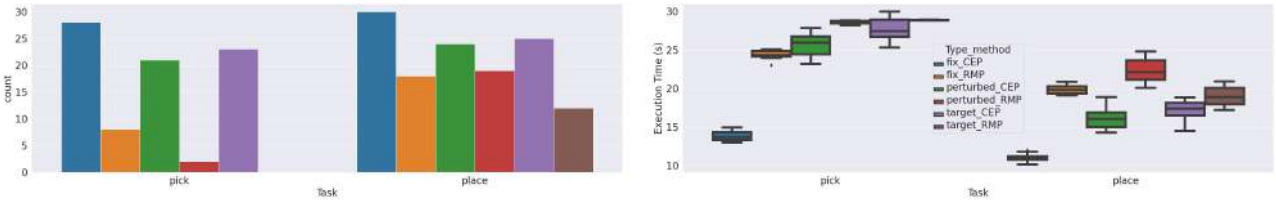


Figure 10. Left: Number of successful picks and places. Right: boxplot showing the execution time to solve the pick and place tasks. We evaluate the performance for (i) fix targets and no disturbances, (ii) under human physical disturbances and, (iii) under target modifications.

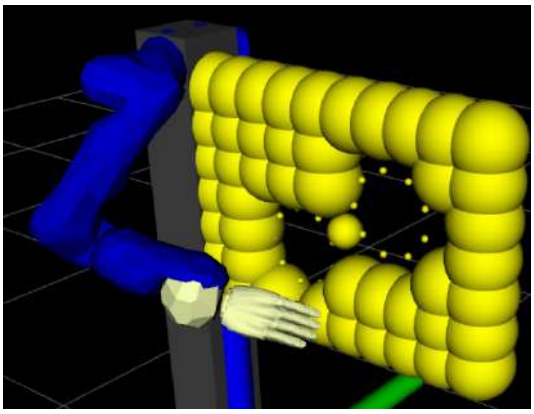


Figure 11. A visual representation of the sphere-based collision body for the real robot experiments wall. The collision body is composed of 67 spheres with different radius.

We can also observe the increase in execution time under human perturbances and target modifications for both picking and placing tasks. Due to the stochasticity, the human injects in the robot's motion, it requires additional time to solve the task. In the case of human perturbances, the injected noise is by physically stopping and moving the robot around. Additionally, the robot might be set in an uncomfortable configuration and it requires additional time to recover and solve the task. In the case of the target modifications, the additional execution time is usually due to the human lack of steadiness and sensor errors. The robot tries to grasp the object once a certain distance threshold to the object is passed. Given the sensor disturbances and the human's lack of steadiness, the robot might require additional time to reduce the threshold distance and pick the object. We can observe, that the execution times are bigger for the picking case rather than the placing task. Due to the lack of maneuverability in the picking task, the robot usually requires way more time to solve the task. Nevertheless, one is far from the obstacles, the robot's possible movements increases and it can solve the task faster.

Conclusion 3 We evaluated the validity of CEP as a reactive motion generator in a real system. We have observed that the robot is able to reactively adapt to unmodelled perturbances such as human physical disturbances or online target modifications and still solve the pick and place task while moving through a narrow hole. Nevertheless, we observe the locality of CEP in some configurations. The robot might get stuck in a local optimum trying to enter through the hole. Due to this, we suggest integrating learning components or path planning components to be able to consider longer horizon information and resolve the local minima easier.

6.3 Learning with structured policies

In this experimental section, we evaluate the performance of CEP as structured policy in a reinforcement learning problem. We perform the experiments to answer the following questions

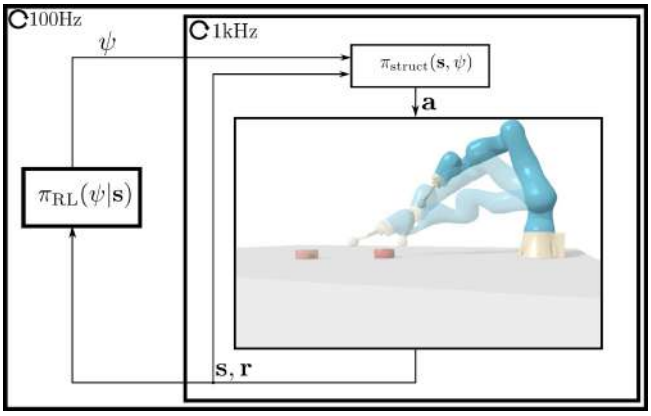


Figure 12. A block diagram of a reinforcement learning problem with a structured policy. The RL agent π_{RL} and the structured policy π_{struct} might run to different control frequencies. Given the current state $\boldsymbol{s}$, the RL policy samples a parameter vector $\boldsymbol{\psi}$. This parameter vector is input in the structured policy and the control action is computed $\boldsymbol{a}$.

Q1: Can we improve the learning performance of the reinforcement learning problem by integrating guiding priors through CEP?

Q2: Can we reduce the number of collisions while learning by integrating obstacle avoidance priors through CEP?

6.3.1 Learning how to hit a puck We consider the problem of learning how to hit a puck and putting it in a certain target position (Figure 12). We consider this task a good experiment to investigate the benefits of integrating priors for learning. A desirable policy should learn to hit the puck without colliding against the table. Nevertheless, given that the puck is close to the table, it is hard to find a policy that weights properly both objectives. We investigate the benefit of CEP as a structured policy to deal with such situations. Additionally, if we consider the whole workspace of the robot, there are very few regions in the state-action space that make the robot move the puck. Most of the possible state-action pairs won't influence the position of the puck. Thus a prior guiding the robot to the puck might be very helpful to explore in more informative regions.

We use a 7 dof LBR-IIWA robot. A visual representation of the task can be found at Figure 12. The reinforcement learning agent π_{RL} receives as input the state $\boldsymbol{s}$. The state $\boldsymbol{s} \in \mathbb{R}^{18}$ is represented by the end-effector's cartesian position $\boldsymbol{x}_{\text{ee}}$, puck's position $\boldsymbol{x}_{\text{puck}}$, their relative position $r_{\text{p-ee}} = \boldsymbol{x}_{\text{ee}} - \boldsymbol{x}_{\text{puck}}$, the puck's velocity $\boldsymbol{v}_{\text{puck}}$, the end effector's velocity $\boldsymbol{v}_{\text{ee}}$ and the target position $\boldsymbol{x}_{\text{target}}$. The output of the reinforcement learning agent is the parameter $\boldsymbol{\psi}$. For our experiment, the output $\boldsymbol{\psi} \in \mathbb{R}^3$ represents the desired task space cartesian velocity in the end-effector. The robot is always initialized with the same joint configuration and each episode last 300 steps (*it lasts 300 steps for the reinforcement learning agent, but 3000 steps for the structured policy* π_{struct}).

Policy Setup The structured policy receives the reinforcement learning agent's output, $\boldsymbol{\psi}$, and the state $\boldsymbol{s}$ and computes the desired configuration space acceleration as action $\boldsymbol{a} = \ddot{\boldsymbol{q}}$.

We consider two baselines as structured policies: direct operational space control (Khatib 1987) and residual operational space control (Silver et al. 2018; Johannink et al. 2019). The structured policy for direct operational space control is modelled by

$$\pi_{\text{struct}}(\boldsymbol{s}, \boldsymbol{\psi}) = \boldsymbol{J}^{\dagger} \boldsymbol{K}(\boldsymbol{\psi} - \dot{\boldsymbol{q}}), \tag{65}$$

with $\boldsymbol{J}^{\dagger}$ the Jacobian pseudoinverse of the forward kinematics to the end-effector, $\dot{\boldsymbol{q}}$ the current robot joint configuration and $\boldsymbol{K}$ a damping gain. The controller defines a velocity error correction in the task space. Then, the desired task space acceleration is map to the configuration space by the Jacobian pseudoinverse. The residual operational space controller is modelled by

$$\begin{aligned} \pi_{\text{struct}}(\boldsymbol{s}, \boldsymbol{\psi}) &= \boldsymbol{J}^{\dagger} \boldsymbol{K}(\dot{\boldsymbol{q}}^{*} - \dot{\boldsymbol{q}}) \\ \dot{\boldsymbol{q}}^{*} &= \boldsymbol{\psi} + \pi_g(\boldsymbol{s}), \end{aligned} \tag{66}$$

with $\pi_g(\boldsymbol{s})$ the guiding policy. Residual task space control applies a similar approach to direct operational space control. Nevertheless, the desired task space velocity is defined by the linear sum of $\boldsymbol{\psi}$ and $\pi_g(\boldsymbol{s})$. $\pi_g(\boldsymbol{s})$ is a guiding policy. In our experiments, we model $\pi_g(\boldsymbol{s})$ as a CEP with an obstacle avoidance energy to the table and a puck attractor energy. Thus, the desired velocity is represented as the linear sum of the reinforcement learning action and the CEP action.

We compare these baselines with respect to using CEP as the structured policy (Section 5). Our model policy is modelled by

$$\pi_{\text{struct}}(\boldsymbol{s}, \boldsymbol{\psi}) = \arg\max_{\ddot{\boldsymbol{q}}} \log \left(\prod\nolimits_k \pi_k(\ddot{\boldsymbol{q}}|\boldsymbol{s}, \boldsymbol{\psi}) \right). \tag{67}$$

In contrast with the residual policy, which linearly combines the output of the reinforcement learning agent and the CEP, in our case, we input the action of the reinforcement learning agent $\boldsymbol{\psi}$ as an additional parameter to condition the energy policies. Then, the optimal action is computed between the conditioned energy policies and the priors. The CEP is built by three energy policy components:

- A target (puck position) reaching energy in the end-effector;
- An obstacle avoidance energy in the end-effector (to avoid collisions against the table);
- $\boldsymbol{\psi}$ parameterized velocity tracking energy in the end-effector.

The chosen residual policy combines the output of the reinforcement learning agent $\boldsymbol{\psi}$ and the output of a defined CEP, π_g linearly. In contrast, the CEP policy takes as input parameter the reinforcement learning agent's output $\boldsymbol{\psi}$ and solves the maximization problem combining a $\boldsymbol{\psi}$ parameterized policy and two defined policies.

Both baselines have been widely applied as structured policies in reinforcement learning (Iriondo et al. 2019; Johannink et al. 2019; Lee et al. 2019; Schaff and Walter 2020; Funk et al. 2021). We choose an operational space control baseline to investigate the benefit of the target reaching bias terms in the CEP. Given both CEP and residual policy uses an attractor to the target, we aim to investigate if it provides an additional benefit with respect

to not considering an attractor bias in the policy. We choose residual control to investigate the benefit of the obstacle avoidance prior in CEP. Even if both consider an obstacle avoidance prior, the prior and the reinforcement learning action are integrated differently. We aim to investigate the benefits and perks of imposing obstacle avoidance in our method concerning the residual control method.

Problem Setup To properly investigate the benefit of CEP, we extend the previous work in (Urain et al. 2021) with two additional MDPs. The three MDPs consider the same transition dynamics model, but they differ in the reward function. The three rewards contain a reward signal that defines the task to solve (move the puck to the target). The three of them differ in the inductive biases we additionally integrate into the reward function. We aim to study the influence of reward shaping with respect to the influence of structured policies to improve the learning performance.

The first reward is composed of the distance between the end-effector and the puck and distance between the puck and the target:

$$r_1 = -d_{\text{ee-puck}} - d_{\text{puck-target}}.$$

The distance between the end-effector and the puck is an inductive bias that helps the agent to find a policy that hits the puck. A similar inductive bias is integrated into the residual and CEP policies with the attractor policy.

The second reward function additionally considers a negative terminal cost if the robot hits the table, $r_T = -100$.

$$r_2 = -d_{\text{ee-puck}} - d_{\text{puck-target}} + r_T(\boldsymbol{s}_T).$$

The negative terminal cost is an inductive bias that pushes the robot to avoid the table. If the robot collides against the table, we stop the episode and add the terminal cost.

Finally, the last reward function (the one in (Urain et al. 2021)), considers only the reward distance between the puck and the target

$$r_3 = -d_{\text{puck-target}} + r_T(\boldsymbol{s}_T).$$

Eliminating the attractor to the puck, the direct operational space controller lacks any source of bias to approximate to the puck. We aim to study if the attractor inductive bias might have any influence on our learning performance. We claim that CEP can be used with any arbitrary reinforcement learning algorithm. To investigate its performance, we have conducted the experiments with several deep reinforcement learning algorithms (PPO (Schulman et al. 2017), SAC (Haarnoja et al. 2018b), DDPG (Lillicrap et al. 2015), TD3 (Fujimoto et al. 2018)) implemented in MushroomRL (D'Eramo et al. 2021).

Comparative evaluation We investigate the learning performance of the three structured policies in terms of the accumulated reward and accumulated collisions against the table per epoch. We present the obtained results in Figure 13. We additionally present a visualization of the learned policies for every case in Figure 7 (Extension 7).

We start evaluating the results for the first reward (Figure 13 top row). Observing the discounted reward, we can see that the three controllers can achieve a similar performance (direct operational space control is slightly worse). While CEP and residual policies initial return is close to the prior, the direct operational space control starts with a worse policy. Nevertheless, given that we have added a distance reward to the puck, the direct controller is able to get close to CEP and residual in a few episodes. The performance is pretty different if we observe the collisions. The collision avoidance prior allows CEP to explore without a single collision. In contrast, residual and direct controllers collide quite often while exploring. The residual controller has the obstacle avoidance prior encoded in the guiding policy. Nevertheless, we can observe that the prior is not strong enough and the robot collides quite often. In the third column, we present the performance of CEP for different deep reinforcement learning algorithms. As we can observe, the agent is able to solve the problem with any learning algorithm.

In the second experiment, we additionally add a terminal cost if the robot collides with the table. We present the results in Figure 13 middle row. We can observe that while CEP maintains a similar learning curve than in the first problem, the learning curve for residual learning and direct control slows down. In the current MDP, the robot not only needs to find a policy to hit the puck but also, avoid collisions against the table. We hypothesize that given CEP is able to impose the table collision properly, it will not explore in state-space regions that lead to a collision. Therefore, the learning agent is completely focused on the problem of hitting the puck. On the contrary, direct and residual learning policies are constantly hitting against the table. Due to this, the reinforcement learning agent requires to learn a policy that both avoids collisions and also hits the puck. This hypothesis match with the collisions plot. Both direct and residual policies have multiple collisions in the first episodes. Nevertheless, during the learning, they find policies that are able to avoid them. In the case of CEP, the obstacle avoidance prior is sufficiently strong to have an obstacle-free learning process. Similar to the first problem, all the reinforcement learning algorithms can solve the problem with similar performances.

Finally, in the third experiment, we eliminate from the reward function the prior that guides the robot close to the puck. In our previous experiments, the guiding prior is included in both the reward signal and the structured policies. We aim to investigate the learning performance when we impose the guiding attractor only in the structured policies. The obtained results are presented in Figure 13 bottom row. The performance of our method remains good even if the inductive bias is eliminated from the reward. The inductive bias in the policy is enough to find a good behavior to hit the puck. We can observe that also residual policy learning is performing well, similarly to the second environment. Nevertheless, the direct controller is not able to solve the problem and the performance decays from previous experiments. The reward function is highly sparse (the robot receives information only if it hits the puck) and the direct control lacks any guiding inductive bias to be close to the puck. In conclusion, a lack of guided exploration makes the controller not find proper behavior.

Conclusion 4 In this experiment we investigate the influence of integrating inductive biases in a robot reinforcement learning problem. We include two inductive

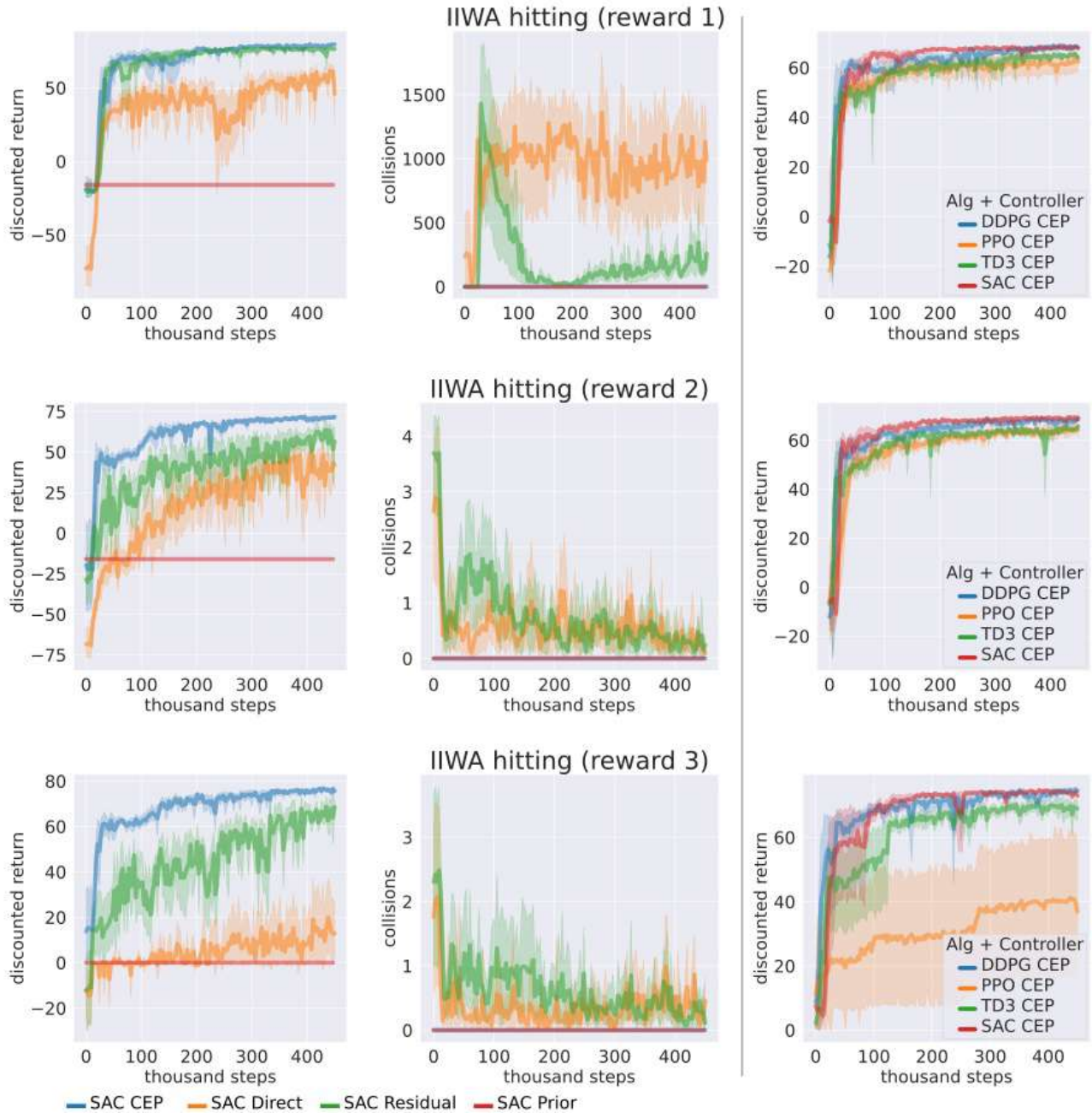


Figure 13. Obtained results for the hitting a puck experiment. Column 1 and 2 present a comparison between different structured policies. Column 3 presents a comparison between different reinforcement learning algorithms.

biases: a collision-avoidance bias to encourage the robot not to collide against the table and an attractor bias that encourages the robot to get close to the puck. We investigate the differences between integrating the inductive biases directly in the reward or integrating them in the policy. Additionally, we investigate the differences linearly sum inductive biases (residual controller) or through an optimization problem (composable energy policies). A relevant conclusion is that in contrast with other structured policies, only CEP is able to guarantee a learning without a single collision. the collision avoidance bias in the residual controller does not impose sufficient constraints. On the other hand, the collision avoidance bias in the reward function requires the robot to collide against the table to receive the reward signal to learn not to collide. Thus, we conclude that the CEP approach is a relevant approach to guarantee safer exploration. In contrast, we observe that the attractor bias is properly integrated with the residual approach and also through the reward function. While CEP is able to also integrate this inductive bias, we do not see any major benefit in our approach with respect to others.

7 Related work

In the main part of this article, we focus on modeling reactive motion generators. Nevertheless, reactive motion generators have been widely explored for both robot local navigation and robot reinforcement learning. In this section, we want to briefly summarize the existing work on both topics.

Multi-objective reactive motion generation The problem of reactively generating motion for local robot navigation

satisfies three conditions: (i) there are multiple objectives that must be jointly satisfied, (ii) usually, these objectives are defined in arbitrary task spaces and (iii) we should be able to compute the solution fast enough to have high control frequencies and be reactive. Two of the earliest solutions to this problem are proposed in (Khatib 1985, 1987). In (Khatib 1985), artificial potential fields method is proposed. This method provides a solution for combining multiple obstacle avoidance objectives as a combination of repulsive potential fields. In (Khatib 1987), the idea of operational space control is defined. Operational space control provides a method for mapping the desired policies defined in the task space to the configuration space in which the robot is controlled.

Inspired by these early efforts, in (Ratliff et al. 2018), Riemannian motion policies are introduced. In this work, the problem of properly combining policies defined in different task spaces is addressed. Riemannian motion policies focus on the concept of the metric to properly weight the contribution of each policy in the final action. Later works (Cheng et al. 2018; Li et al. 2021) improve the riemannian motion policies in terms of computational efficiency. There has been a set of alternative works, dealing with different problems in Riemannian motion policies. In (Shaw et al. 2021) pullback bundle dynamical systems are proposed. The work proposes a method to combine multiple policies defined in non-Euclidean spaces. Geometric Fabrics (Ratliff et al. 2020, 2021; Xie et al. 2020) propose to model the policy composition in terms of Finsler geometries. As shown in their work, modelling the problem in terms of Finsler geometries, they can easily guarantee stable behaviors in contrast to Riemannian motion policies.

All of the above methods assume the composition of a set of myopic controllers. In a different direction, some researchers have tried to study the composition of policies that are optimised to solve a longer horizon control problem. In (Todorov 2009), linearly-solvable Markov Decision Processes (LMDP) is presented. As shown in the work, a weighted sum of individually optimal policies was proven to be the optimal control problem solution for a reward function defined as weighted-sum of individual rewards and differing only in the terminal reward. In a similar vein (Haarnoja et al. 2018a; Tasse et al. 2020) proposes suming a set of optimal Q function. Additionally, in (Haarnoja et al. 2018a) the distance between the optimal Q function and the sum of Q functions is investigated.

Finally, a set of works propose solving the multi-objective reactive motion generation problem by numerical optimization. Dynamic Window Approach (DWA) (Fox et al. 1997; Van Den Berg et al. 2011) solves the reactive motion generation for a 2D planar robot in a two steps optimization algorithm. First, the search space of possible actions is reduced given a set of constraints. Then, given an objective function, the optimal action is selected. Model predictive control (MPC) methods (Garcia et al. 1989; Erez et al. 2013; Dentler et al. 2016) consider a non-myopic trajectory optimization problem to find reactively the optimal action satisfying multiple objectives. To reduce the computational requirements, MPC methods initialize the optimization problem with the previously computer solution. These methods usually assume simplified kinematics and dynamics models and quadratic cost functions to be computationally efficient and be reactive.

Our work lays in a middle-ground. We consider a myopic (one-step ahead) optimization problem to find the optimal action similarly to artificial potential fields and Riemannian motion policies. Nevertheless, we do not assume there exist an analytic expression for our solution and rather we solve a numerical optimization problem in every control step as in model predictive control.

Structured policies in robot reinforcement learning

Integrating inductive biases into the reinforcement learning problem has been shown to be effective in improving the learning performance of the reinforcement learning agents. These inductive biases are usually integrated into the problem through the reward function (*reward shaping*) or through the policy (*structured policy*). Structured policies modify the action space of the reinforcement learning agent. Rather than sampling actions that directly influence the robot such as torques, the reinforcement learning agent samples actions in a parameter space. These parameters are later inputted into the structured policy and the robot control signal is computed.

Structured policies have a long history in robot reinforcement learning. Operational space controllers (Peters and Schaal 2007; Lee et al. 2019) transform the action space from the configuration space to the task space. It is shown that the robot is able to learn better policies if the reward function is also defined in the task space. A big set of works have considered applying reinforcement learning in the parameter space of movement primitives such as DMP or handcrafted primitives (Mülling et al. 2013; Daniel et al. 2016; Pertsch et al. 2021; Bahl et al. 2020; Dalal et al. 2021). Considering a DMP as structured policy guarantees inductive biases such as stability or smoothness in the robot's behavior. Residual Policy Learning approaches (Silver et al. 2018; Johannink et al. 2019), model the structured policy by a linear sum of the reinforcement learning action and the output of a guiding policy. Assuming the guiding policy is properly modeled for the task, the reinforcement learning agent is expected to learn the residuals of the guiding policy and improve the performance of the guiding policy.

A different approach is proposed in (Li et al. 2021). Instead of splitting the policy between the reinforcement learning agent and the structured policy, they propose to model the reinforcement learning policy directly as a RMP. The action to apply in the robot is directly sampled from the agent, but the agent model is a parameterized RMP. They claim that both the policy leaves and the task maps can be parameterized and learned.

In contrast with previous works that assume an explicit structured policy; in our work, we propose to model the structured policy via an optimization function. The reinforcement learning agent samples a set of actions that parameterize the energy policies. Then, we apply an optimization problem to find the optimal action. As shown in the experiments, this optimization problem can easily impose hard constraints (through uniform distributions) and guarantee safer learning.

8 Discussion

CEP can be viewed as a generalization on RMP where the cost functions are not limited to be quadratic. This generalization provides additional flexibility to model the policy components and find better alternatives to represent the policy components. From a different perspective, CEP can be viewed as a particular case of MPC, in which the control horizon is fixed to a single step. Fixing the control horizon to a single step allows us to reduce the optimization variables and find control actions for a high-dimensional robot and a set of cost functions with a low computational budget. We show empirically, that with a proper choice of energies, one-step ahead optimization can still solve complex cluttered environments without the computational requirements of optimizing over longer horizons as in MPC. Nevertheless, it is important to remark that our approach is myopic and thus, we will never have guarantees of satisfying all the objectives in the long run. To have the guarantees of solving a long-horizon problem, we might instead rely on long-horizon planning methods. Additionally, we embrace a probabilistic interpretation of the multi-objective reactive motion generation problem. Framing the problem in a probabilistic view allow us to build connections between the literature in Bayesian inference (Jaynes 1957; Rawlik et al. 2012) or in energy based composition (Hinton 2002; Du et al. 2020) and the literature in reactive motion generation (Khatib 1987; Ratliff et al. 2018). We consider that this probabilistic interpretation is beneficial to integrate learning components in the reactive motion generation problem. From this view, we can built policy component as maximum likelihood distributions for a given dataset or maximum-entropy policies for a given reward.

Introduced in Section 3, RMP can be viewed as the solution of a myopic control as inference problem in which each component is modelled by a normal distribution. A normal distribution assumes that the action distribution is unimodal. There exist an optimal action (the mean) to satisfy a particular objective and the quality of the rest of the actions is measured given a Mahalanobis distance to the optimal one. While framing the problem in terms of normally distributed policy components have some benefits (there exist a close form solution for the optima); the expressivity might be limited for other components. We have shown experimentally, that uniformly distributed policies might be more benefitial to represent obstacle avoidance policies. Nevertheless, a drawback of considering non-quadratic policies is that we lack an analytical solution of our optimization problem and we require to use stochastic optimization algorithms to find the optimal action. Through a set of ablation studies, we have evaluated the computational requirements of our method and find out that with the current hardware, we can achieve control frequencies up to 500Hz with a non optimized code. We expect that with a highly optimized code and with more powerful hardware, we could achieve 1kHz control frequencies.

In our work, we have additionally evaluate the performance of CEP as a structured policy. In reinforcement learning, high dimensional state-action spaces with sparse rewards might require an excessive amount of samples to find a proper behavior. Additionally, the robot might have undesirable collisions while exploring. A common approach to deal with this problem is by exploring in the parameter space of a low-level controller. Through this abstraction we can impose all the desired inductive bias in the low-level controller and explore in manifold generated by the inductive biases. In contrast with most of the structured policies in the literature, in CEP, the structured policy is represented by an optimization problem over a set of parameterized implicit functions. The reinforcement learning agent samples a set of parameters that conditions the objective function to optimize. Through this abstration we can combine inductive bias costs and reinforcement learning conditioned costs and solve an optimization problem that aims to satisfy all the objectives. Through this approach, imposing collision avoidance constraints or guiding bias is simple as we only require to agregate an additional energy function to the objective. It also provides a modular learning as the reinforcement learning conditioned policy can be completely independent from the inductive bias policies.

9 Conclusion and future work

We have introduced a probabilistic approach for multi-objective reactive motion generation. We have shown theoretically the relations between CEP and RMP and observe that the increase in flexibility when modelling the policies could improve the composability performance with respect to RMP or APF methods. CEP is a general framework that allows the composition of multiple sources policies defined in abritrary task spaces.

When integrated as a structured policy for reinforcement learning, we have shown that CEP provides a novel learning structure. The reinforcement learning agent samples a set of parameters that are integrated as conditioning elements of an objective function and the optimal action is computed by solving this low level optimization problem. This bi-level optimization problem allows the integration of safety constraints through uniformly distributed inductive biases or guiding inductive biases without changing the policy model, but simply including additional components to the objective function. We have shown experimentally that the implicit structured approach provides higher safety guarantees with respect to explicit structured policies.

A missing element in the current work is observing the performance of CEP with data-driven policy components. All the components we have considered are analitically computed policies or reinforcement learning based learned policies. We consider as future work, exploring the composition of multiple policies learned by imitation learning. Combining CEP architecture with imitation learning allows the composition of multiple demonstrations given in different task spaces. This would open the possibility of composing expert demonstrations from multiple sources and in different state-action spaces in a single policy. We expect this approach to increase the generalization and modularity properties in robot learning.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programmes under grant agreement No. #820807 (SHAREWORK).

The Authors declare that there is no conflict of interest.

References

Aljalbout E, Chen J, Ritt K, Ulmer M and Haddadin S (2021) Learning vision-based reactive policies for obstacle avoidance. In: *Conference on Robot Learning*. PMLR, pp. 2040–2054.

Attias H (2003) Planning by probabilistic inference. In: *International Workshop on Artificial Intelligence and Statistics*. PMLR, pp. 9–16.

Bahl S, Mukadam M, Gupta A and Pathak D (2020) Neural dynamic policies for end-to-end sensorimotor learning. *arXiv preprint arXiv:2012.02788* .

Bhardwaj M, Sundaralingam B, Mousavian A, Ratliff ND, Fox D, Ramos F and Boots B (2022) Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In: Faust A, Hsu D and Neumann G (eds.) *Proceedings of the 5th Conference on Robot Learning*, *Proceedings of Machine Learning Research*, volume 164. PMLR, pp. 750–759. URL `https://proceedings.mlr.press/v164/bhardwaj22a.html`.

Botev ZI, Kroese DP, Rubinstein RY and L'Ecuyer P (2013) The cross-entropy method for optimization. In: *Handbook of statistics*, volume 31. Elsevier, pp. 35–59.

Bylard A, Bonalli R and Pavone M (2021) Composable geometric motion policies using multi-task pullback bundle dynamical systems. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7464–7470.

Carpentier J, Saurel G, Buondonno G, Mirabel J, Lamiraux F, Stasse O and Mansard N (2019) The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In: *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, pp. 614–619.

Cheng CA, Mukadam M, Issac J, Birchfield S, Fox D, Boots B and Ratliff N (2018) Rmpflow: A computational graph for automatic motion policy generation. In: *International Workshop on the Algorithmic Foundations of Robotics*.

Da Silva M, Durand F and Popović J (2009) Linear bellman combination for control of character animation. *ACM SIGGRAPH* .

Dalal M, Pathak D and Salakhutdinov RR (2021) Accelerating robotic reinforcement learning via parameterized action primitives. *Advances in Neural Information Processing Systems* 34.

Daniel C, Neumann G, Kroemer O and Peters J (2016) Hierarchical relative entropy policy search. *Journal of Machine Learning Research* 17: 1–50.

De Boer PT, Kroese DP, Mannor S and Rubinstein RY (2005) A tutorial on the cross-entropy method. *Annals of operations research* 134(1): 19–67.

Dentler J, Kannan S, Mendez MAO and Voos H (2016) A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors. In: *2016 IEEE conference on control applications (CCA)*. IEEE, pp. 519–525.

D'Eramo C, Tateo D, Bonarini A, Restelli M and Peters J (2021) Mushroomrl: Simplifying reinforcement learning research. *Journal of Machine Learning Research* 22(131): 1–5.

Du Y, Li S and Mordatch I (2020) Compositional visual generation and inference with energy based models. In: *Conference on Neural Information Processing Systems*.

Erez T, Lowrey K, Tassa Y, Kumar V, Kolev S and Todorov E (2013) An integrated system for real-time model predictive control of humanoid robots. In: *2013 13th IEEE-RAS International conference on humanoid robots (Humanoids)*. IEEE, pp. 292–299.

Florence P, Lynch C, Zeng A, Ramirez OA, Wahid A, Downs L, Wong A, Lee J, Mordatch I and Tompson J (2022) Implicit behavioral cloning. In: *Conference on Robot Learning*. PMLR, pp. 158–168.

Fox D, Burgard W and Thrun S (1997) The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1): 23–33.

Fujimoto S, Hoof H and Meger D (2018) Addressing function approximation error in actor-critic methods. In: *International conference on machine learning*. PMLR, pp. 1587–1596.

Funk N, Schaff C, Madan R, Yoneda T, Urain J, Watson J, Gordon EK, Widmaier F, Bauer S, Srinivasa SS, Bhattacharjee T, Walter MR and Peters J (2021) Benchmarking structured policies and policy optimization for real-world dexterous object manipulation. *IEEE Robotics and Automation Letters* 7(1): 478–485.

Garcia CE, Prett DM and Morari M (1989) Model predictive control: Theory and practice—a survey. *Automatica* 25(3): 335–348.

Ge SS and Cui YJ (2002) Dynamic motion planning for mobile robots using potential field method. *Autonomous robots* 13(3): 207–222.

Gibbs JW (1902) *Elementary principles in statistical mechanics: developed with especial reference to the rational foundations of thermodynamics*. C. Scribner's sons.

Haarnoja T, Pong V, Zhou A, Dalal M, Abbeel P and Levine S (2018a) Composable deep reinforcement learning for robotic manipulation. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 6244–6251.

Haarnoja T, Tang H, Abbeel P and Levine S (2017) Reinforcement learning with deep energy-based policies. In: *International Conference on Machine Learning*. PMLR, pp. 1352–1361.

Haarnoja T, Zhou A, Abbeel P and Levine S (2018b) Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: *International conference on machine learning*. PMLR, pp. 1861–1870.

Hinton GE (2002) Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8): 1771–1800.

Iriondo A, Lazkano E, Susperregi L, Urain J, Fernandez A and Molina J (2019) Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning. *Applied Sciences* 9(2): 348.

Jaynes ET (1957) Information theory and statistical mechanics. *Physical review* 106(4): 620.

Johannink T, Bahl S, Nair A, Luo J, Kumar A, Loskyll M, Ojea JA, Solowjow E and Levine S (2019) Residual reinforcement learning for robot control. In: *International Conference on Robotics and Automation*. IEEE, pp. 6023–6029.

Kaelbling LP and Lozano-Pérez T (2011) Hierarchical task and motion planning in the now. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1470–1477.

Kaelbling LP and Lozano-Pérez T (2013) Integrated task and motion planning in belief space. *The International Journal of Robotics Research* 32(9-10): 1194–1227.

Kalakrishnan M, Chitta S, Theodorou E, Pastor P and Schaal S (2011) Stomp: Stochastic trajectory optimization for motion planning. In: *IEEE international conference on robotics and automation*. IEEE, pp. 4569–4574.

Kappler D, Meier F, Issac J, Mainprice J, Cifuentes CG, Wüthrich M, Berenz V, Schaal S, Ratliff N and Bohg J (2018) Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters* 3(3): 1864–1871.

Kavraki LE, Svestka P, Latombe JC and Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12(4): 566–580.

Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: *IEEE International Conference on Robotics and Automation*, volume 2. pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

Khatib O (1987) A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation* .

Lambert A, Ramos F, Boots B, Fox D and Fishman A (2021) Stein variational model predictive control. In: *Conference on Robot Learning*. PMLR, pp. 1278–1297.

LaValle SM (2006) *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press.

LaValle SM and Kuffner JJ (2001) Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions* 5: 293–308.

Lee MA, Zhu Y, Srinivasan K, Shah P, Savarese S, Fei-Fei L, Garg A and Bohg J (2019) Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 8943–8950.

Levine S (2018) Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909* .

Li A, Cheng CA, Rana MA, Xie M, Van Wyk K, Ratliff N and Boots B (2021) RMP2: A Structured Composable Policy Class for Robot Learning. In: *Robotics: Science and Systems (R:SS)*.

Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D and Wierstra D (2015) Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* .

Lutter M, Mannor S, Peters J, Fox D and Garg A (2021) Value iteration in continuous actions, states and time. In: *International Conference on Machine Learning*. PMLR, pp. 7224–7234.

Morari M and Lee JH (1999) Model predictive control: past, present and future. *Computers & Chemical Engineering* 23(4-5): 667–682.

Mukadam M, Dong J, Yan X, Dellaert F and Boots B (2018) Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research* 37(11): 1319–1340.

Mülling K, Kober J, Kroemer O and Peters J (2013) Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research* 32(3): 263–279.

Ohtsuka T (2004) A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica* 40(4): 563–574.

Paraschos A, Daniel C, Peters JR and Neumann G (2013) Probabilistic movement primitives. In: *Advances in Neural Information Processing Systems*. pp. 2616–2624.

Park JJ, Florence P, Straub J, Newcombe R and Lovegrove S (2019) Deepsdf: Learning continuous signed distance functions for shape representation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 165–174.

Peng XB, Chang M, Zhang G, Abbeel P and Levine S (2019) Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems 32* .

Pertsch K, Lee Y and Lim J (2021) Accelerating reinforcement learning with learned skill priors. In: *Conference on Robot Learning*. PMLR, pp. 188–204.

Peters J and Schaal S (2007) Reinforcement learning by reward-weighted regression for operational space control. In: *Proceedings of the 24th international conference on Machine learning*. pp. 745–750.

Poignet P and Gautier M (2000) Nonlinear model predictive control of a robot manipulator. In: *6th International workshop on advanced motion control. Proceedings (Cat. No. 00TH8494)*. IEEE, pp. 401–406.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) Chomp: Gradient optimization techniques for efficient motion planning. In: *IEEE International Conference on Robotics and Automation*. pp. 489–494.

Ratliff ND, Issac J, Kappler D, Birchfield S and Fox D (2018) Riemannian motion policies. *arXiv preprint arXiv:1801.02854* .

Ratliff ND, Van Wyk K, Xie M, Li A and Rana MA (2020) Optimization fabrics. *arXiv preprint arXiv:2008.02399* .

Ratliff ND, Van Wyk K, Xie M, Li A and Rana MA (2021) Generalized nonlinear and finsler geometry for robotics. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 10206–10212.

Rawlik K, Toussaint M and Vijayakumar S (2012) On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII* .

Schaff C and Walter MR (2020) Residual policy learning for shared autonomy. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Schulman J, Duan Y, Ho J, Lee A, Awwal I, Bradlow H, Pan J, Patil S, Goldberg K and Abbeel P (2014) Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research* 33(9): 1251–1270.

Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O (2017) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* .

Shaw S, Abbatematteo B and Konidaris G (2021) Rmps for safe impedance control in contact-rich manipulation. *arXiv preprint*

arXiv:2109.12103 .

Silver T, Allen K, Tenenbaum J and Kaelbling L (2018) Residual policy learning. *arXiv preprint arXiv:1812.06298* .

Sola J, Deray J and Atchuthan D (2018) A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537* .

Sutton RS, Precup D and Singh S (1999) Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2): 181–211.

Tasse GN, James S and Rosman B (2020) A boolean task algebra for reinforcement learning. *Conference on Neural Information Processing Systems* .

Todorov E (2009) Compositionality of optimal control laws. *Advances in neural information processing systems* 22: 1856–1864.

Toussaint M (2009) Robot trajectory optimization using approximate inference. In: *international conference on machine learning*. pp. 1049–1056.

Urain J, Ginesi M, Tateo D and Peters J (2020) Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5231–5237.

Urain J, Li A, Liu P, D'Eramo C and Peters J (2021) Composable energy policies for reactive motion generation and reinforcement learning. In: *Robotics: Science and Systems (R:SS)*.

Van Den Berg J, Guy SJ, Lin M and Manocha D (2011) Reciprocal n-body collision avoidance. In: *Robotics research*. Springer, pp. 3–19.

Van Niekerk B, James S, Earle A and Rosman B (2019) Composing value functions in reinforcement learning. In: *International Conference on Machine Learning*. pp. 6401–6409.

Williams G, Aldrich A and Theodorou EA (2017) Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics* 40(2): 344–357.

Xie M, Van Wyk K, Li A, Rana MA, Wan Q, Fox D, Boots B and Ratliff N (2020) Geometric fabrics for the acceleration-based design of robotic motion. *arXiv preprint arXiv:2010.14750* .

Ziebart BD, Bagnell JA and Dey AK (2010) Modeling interaction via the principle of maximum causal entropy. In: *International Conference on Machine Learning*.

A Index to Multimedia Extensions

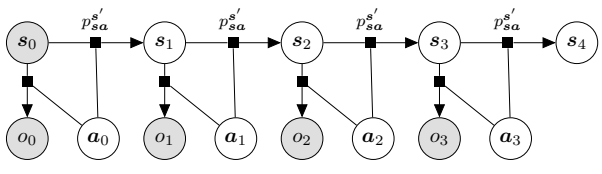
Extension	Media Type	Description
1	Video	Visual Representation of reaching through a cluttered environment
2	Video	Visual Representation of the experiments for real robot reactive pick and place task I: Multiple Initial configurations
3	Video	Visual Representation of the experiments for real robot reactive pick and place task II: Human disturbances
4	Video	Visual Representation of the experiments for real robot reactive pick and place task III: Online picking changes
5	Video	Visual Representation of the experiments for real robot reactive pick and place task IV: Online target changes
6	Video	Visual Representation of the experiments for real robot reactive pick and place task V: Failure cases
7	Video	Visual Representation of the results for the learned policies for the robot hitting a puck

Table 7. Table of Multimedia Extensions

B A Control as Inference view for Composable Energy Policies

In the following section, we want to highlight the connections between composable energy policies and control as inference to evaluate the optimality guarantees of composing energies in a multi-objective optimal control problem.

Figure 14. Graphical model for the Optimal Control problem. $\boldsymbol{s}_t$ denoted the state, $\boldsymbol{a}_t$ denotes the action and o_t is an additional variable representing the optimality of the state and action for a given reward.



We frame optimal control as a Bayesian inference problem (Rawlik et al. 2012; Levine 2018) over the sequence of actions $\boldsymbol{a}_{0:T}$. The optimal control problem is visualized as a graphical model in Fig. 14. The problem is formulated introducing an auxiliary variable $o_{0:T}$ that represents the optimality of $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$ under a certain reward function, $p(o_t|\boldsymbol{s}_t, \boldsymbol{a}_t) \propto \exp(r(\boldsymbol{s}_t, \boldsymbol{a}_t))$. Given a certain prior distribution $q(\boldsymbol{a}|\boldsymbol{s}_0)$ and given $\boldsymbol{s}_0$ is known, the inference problem is

$$p(A_0^T|\boldsymbol{s}_0, O_0^T) = \frac{p(O_0^T|A_0^T, \boldsymbol{s}_0)q(A_0^T|\boldsymbol{s}_0)}{p(O_0^T|\boldsymbol{s}_0)} \tag{68}$$

with

$$p(O_0^T|A_0^T, \boldsymbol{s}_0) = \int_{\boldsymbol{s}_{1:T}} p(O_0^T|S_0^T, A_0^T)p(S_1^T|A_0^T, \boldsymbol{s}_0)dS_1^T \tag{69}$$

where $A_0^T : \{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_T\}$, $O_0^T : \{o_0, \ldots, o_T\}$ and $S_0^T : \{\boldsymbol{s}_0, \ldots, \boldsymbol{s}_T\}$. The are two main directions to solve the posterior in (68). First, methods that frame the problem as an Hidden Markov Model (HMM) and solve it in an Expectation-Maximization approach. Second, methods that compute the posterior in the trajectory level A_0^T. The first, are computationally demanding as they require several forward and backward message passing to compute the posterior. The second, needs to solve the problem in the trajectory level and thus the dimension of the variables grows linearly with the trajectory length, T.

In our work, we consider solving a one-step-ahead optimal control problem. Rather than solving an optimization problem for a sequence of actions A_0^T, we solve the problem for a single step $\boldsymbol{a}_0$.

We can reframe the control as inference problem as a one-step ahead control problem

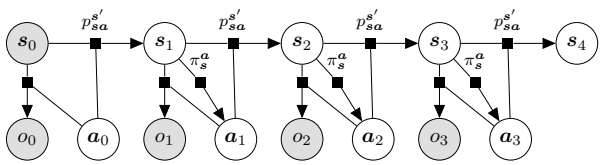
$$p(\boldsymbol{a}_0|\boldsymbol{s}_0, O_0^T) = \frac{p(O_0^T|\boldsymbol{a}_0, \boldsymbol{s}_0)q(\boldsymbol{a}_0|\boldsymbol{s}_0)}{p(O_0^T|\boldsymbol{s}_0)} \tag{70}$$

with

$$p(O_0^T|\boldsymbol{a}_0, \boldsymbol{s}_0) = \int_{S_{1:T}} \int_{A_{1:T}} p(O_0^T|S_0^T, A_0^T)p(S_1^T|\boldsymbol{a}_0, \boldsymbol{s}_0)\pi(A_1^T|S_1^T)dS_1^T dA_1^T. \tag{71}$$

In contrast with the trajectory optimization problem that finds the posterior for the whole trajectory A_0^T, in one-step-ahead control as inference problem, we aim to find the posterior for only the instant next control action, $\boldsymbol{a}_0$. Computing the posterior only for $\boldsymbol{a}_0$ requires the likelihood to be defined as the marginal of not only the state trajectory S_1^T, but also the action trajectory A_1^T. The graphical model for one-step-ahead control as inference is presented in Fig. 15. In one-step ahead control as inference, we introduce an additional distribution π that provides us the probability of A_1^T given S_1^T. This additional policy π is interpreted as the policy the agent will apply in the future. In this context, we are looking for the action the maximizes the cumulative reward in the long horizon trajectory running the policy π.

Figure 15. Graphical model for one-step ahead optimal control problem. In this approach, the actions A_1^T are dependant on S_1^T given a policy π.



(71) can be rewritten as the expectation over $p(O_0^T|S_0^T, A_0^T)$

$$\mathbb{E}_{S_0^T, A_0^T \sim p^\pi(S_0^T, A_0^T|\boldsymbol{s}_0, \boldsymbol{a}_0)} \left[p(O_0^T|S_0^T, A_0^T)\right] \propto \mathbb{E}_{S_0^T, A_0^T \sim p^\pi(S_0^T, A_0^T|\boldsymbol{s}_0, \boldsymbol{a}_0)} \left[\exp(\sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t))\right] = \exp(Q_r^\pi(\boldsymbol{s}_0, \boldsymbol{a}_0)) \quad (72)$$

From what follows, the likelihood for our inference problem is proportional to the $\exp(Q)$ defined over a certain policy π. Given a π, the posterior for our inference problem is given by

$$p(\boldsymbol{a}_0|\boldsymbol{s}_0, O_0^T) \propto \exp(Q_r^\pi(\boldsymbol{s}_0, \boldsymbol{a}_0))q(\boldsymbol{a}_0|\boldsymbol{s}_0). \quad (73)$$

The provided Q function depends on π and then, the quality of our reactive motion generator to solve a long horizon optimal control problem directly depends on the quality of π. In the optimal case, for Q^*, the one-step ahead control problem follows the optimal trajectory, even if the optimization is done locally. Nevertheless, in CEP we aim to study the obtained policy in a multi-objective framework. Given we have a set of Q functions, each being optimal for a particular reward; is the sum of the Q functions still optimal for the sum of the rewards?

B.1 Optimality Guarantees

In CEP, we propose to model the Q function as the sum of a set of optimal Q_k^* functions. Instead, we are aware that $Q_\Sigma = \frac{1}{K}\sum_{k=0}^{K} Q_k^*$ is not the optimal function Q^* for the sum of the rewards $r = \frac{1}{K}\sum_{k=1}^{K} r_k$. The closer Q_Σ is from the optimal Q^*, the closer the product of experts policy would be from the optimal policy. In this section, we study, given a certain reward $r = \frac{1}{2}(r_1 + r_2)$, how much the sum of the individual components Q_Σ diverge from the optimal Q^*. In (Levine 2018) is shown, that the optimal Q function for the control as inference problem can be computed by recursively solving the soft-value iteration (Ziebart et al. 2010)

$$Q(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})}\left[V(\boldsymbol{s}')\right]$$
$$V(\boldsymbol{s}) = \log\left(\int_{\mathcal{A}} \exp(Q(\boldsymbol{a}, \boldsymbol{s}))\mathrm{d}\boldsymbol{a}\right). \quad (74)$$

We assume a finite horizon control as inference problem and evaluate how much Q_Σ diverges from $Q*$ when increasing the control horizon

For $t = T$, the optimal Q function for the sum of the rewards is

$$Q^{*T} = \frac{1}{2}(r_1 + r_2). \quad (75)$$

The individual optimal Q functions for each reward are

$$Q_1^{*T} = r_1\,,\; Q_2^{*T} = r_2. \quad (76)$$

Then, the sum of the Q components is given by

$$Q_\Sigma^T = \frac{1}{2}(Q_1^{*T} + Q_2^{*T}). \quad (77)$$

If we compute the distance between the optimal Q^{*T} and the sum of Q's, Q_Σ^T

$$\Delta Q^T = Q^{*T} - Q_\Sigma^T = 0. \quad (78)$$

From (77), for $T = 1$, the sum of the optimal components Q_Σ is equal to the optimal Q.

For longer temporal horizons, the optimal Q is computed by recursively solving the soft Bellman update backward in time. For computing $t = T - 1$,

$$Q^{*T-1}(\boldsymbol{s}, \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}) + \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})}\left[V^{*T}(\boldsymbol{s}')\right]$$
$$V^{*T}(\boldsymbol{s}) = \log\int_{\mathcal{A}} \exp(Q^T(\boldsymbol{a}, \boldsymbol{s})). \quad (79)$$

we do a soft Bellman update. The difference between Q^{*T-1} and Q_Σ^{T-1}

$$\Delta Q^{T-1} = Q^{*T-1} - Q_\Sigma^{T-1} = \mathbb{E}[V^{*T} - V_\Sigma^T] \quad (80)$$

with

$$V_\Sigma^T = \frac{1}{2}(V_1^{*T} + V_2^{*T}). \quad (81)$$

The distance in the value function can be represented as

$$V^{*T} - V_\Sigma^T = \log\frac{\int_{\mathcal{A}}\exp(Q^{*T})}{\int_{\mathcal{A}}(\exp(Q_1^{*T})\int_{\mathcal{A}}\exp(Q_2^{*T}))^{\frac{1}{2}}} = \log\frac{\int_{\mathcal{A}}\exp(\frac{1}{2}Q_1^{*T})\exp(\frac{1}{2}Q_2^{*T})}{(\int_{\mathcal{A}}\exp(Q_1^{*T})\int_{\mathcal{A}}\exp(Q_2^{*T}))^{\frac{1}{2}}}. \quad (82)$$

Then,

$$Q^{*T-1} = Q_\Sigma^{T-1} + \Delta Q^{T-1} = Q_\Sigma^{T-1} + \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})}\left[\log\frac{\int_{\mathcal{A}}\exp(\frac{1}{2}Q_1^{*T})\exp(\frac{1}{2}Q_2^{*T})}{(\int_{\mathcal{A}}\exp(Q_1^{*T})\int_{\mathcal{A}}\exp(Q_2^{*T}))^{\frac{1}{2}}}\right] \quad (83)$$

From (80) and (82), we can obtain the recurrence relation for the distance error

$$\Delta Q^{t-1} = \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})}\left[\log\frac{\int_{\mathcal{A}}\exp(\frac{1}{2}Q_1^{*t})\exp(\frac{1}{2}Q_2^{*t})\exp(\Delta Q^t)}{(\int_{\mathcal{A}}\exp(Q_1^{*t})\int_{\mathcal{A}}\exp(Q_2^{*t}))^{\frac{1}{2}}}\right]. \quad (84)$$

From (82), we can see that if $Q_1^* = Q_2^*$, then $\Delta Q = 0$ and the more they differ, the bigger the distance error to the

optima Q^*. From the obtained results, we can obtain some conclusions.

Similar theoretical studies have been already developed (Haarnoja et al. 2018a; Van Niekerk et al. 2019; Todorov 2009). In Optimal Control, composable optimality guarantees were proven for linear dynamics, by the LMDP approach. In (Todorov 2009; Da Silva et al. 2009), a weighted policy sum was proven to be optimal for the sum of the rewards, as long as the rewards differ only in the terminal reward. In Haarnoja et al. (2018a); Tasse et al. (2020), the optimality of the composition is studied in a maximum entropy reinforcement learning problem.

C Experiments

C.1 Reaching through a cluttered environment

The modular components (reach target, obstacle avoidance and joint limits avoidance) for both baselines APF and RMP were modeled based on Khatib (1985) and Cheng et al. (2018) respectively. The CEP without hand was modelled with the energy policy components introduced in (Urain et al. 2021) and with the parameters In this work, we have extended the experiments with an additional evaluation with the robot hand. We considered the energy policies in Section 4 to model the energy policies. We considered a target reaching policy, a set of obstacle avoidance policies, joint limits avoidance policy and a joint velocity limits policy. We show in table 9 the parameters we consider for the experiments. The reaching target policy is parameterized by the metric $\mathbf{\Lambda}$ that frames the relevance of this component and α that defines the maximum Euclidean distance of the truncated target position. The obstacle avoidance energy policy is parameterized by the desired minimum distance between the surfaces of the body sphere and the obstacle sphere. The joint velocity limits is parameterized by the metric defining the importance of this component. As we can observe, in contrast with (Urain et al. 2020), in the current approach, we have (i) less tuning parameters for each energy component and (ii) there is an intuition in the meaning of each parameter.

Energy Modules	Parameters		
Reach Target	$\boldsymbol{K}_p = 20.$	$\boldsymbol{K}_v = 30.$	$\alpha = 10.$
Obstacle Avoidance	$\gamma = 0.2$	$\alpha = 4.$	$\beta = 0.1$
Joint Limits avoidance	$\gamma = 0.3$	$\alpha = 4.$	$\beta = 0.1$

Table 8. Component parameters for Composable Energy Policies in reaching through cluttered environment (Urain et al. 2021).

Energy Modules	Parameters
Reach target	$\mathbf{\Lambda} = \boldsymbol{I}$, $\alpha = 0.05$
Obstacle avoidance	$\alpha = r_{\text{obstacle}} + r_{\text{body}}$ + 0.01
Joint velocity limits	$\mathbf{\Lambda} = 0.1\boldsymbol{I}$

Table 9. Component parameters for Composable Energy Policies in reaching through cluttered environment with hand. r_{obstacle} and r_{body} represent respectively, the radius of the collision spheres in the obstacle and the robot body.

C.2 Learning to hit a puck

In the reinforcement learning experiment, we consider the same hyperparameters for CEP, residual operational space control and direct operational space control. Additionally, we keep the same hyperparameters for the three reward functions. We consider the reinforcement learning algorithms implemented in Mushroom-RL (D'Eramo et al. 2021). In the following, we introduce a table with the hyperparameters for PPO, SAC, DDPG and TD3.

Hyperparameters	
policy net	18-128-128-3
policy batch	64
policy learn rate	3e-4
critic learn rate	3e-4
critic net	18-128-128-1
critic batch	256
critic learn rate	3e-4
n_steps_per_fit	600
discount factor	.99
eps_ppo	0.1

Table 10. PPO hyperparameters for hitting the puck

Hyperparameters	
policy net mean	18-128-128-3
policy net sigma	18-128-128-3
policy batch	64
policy learn rate	3e-4
critic learn rate	3e-4
critic net	18-128-128-1
critic batch	256
critic learn rate	3e-4
n_steps_per_fit	1
discount factor	.99
target entropy	-6
warmup transitions	10000
max replay size	200000

Table 11. SAC hyperparameters for hitting the puck

Hyperparameters	
policy net	18-128-128-3
policy batch	64
policy learn rate	3e-4
critic learn rate	3e-4
critic net	18-128-128-1
critic batch	256
critic learn rate	3e-4
n_steps_per_fit	1
discount factor	.99
warmup transitions	10000
max replay size	200000

Table 12. DDPG and **TD3** hyperparameters for hitting the puck