Composable Energy Policies for Reactive Motion Generation and Reinforcement Learning

Julen Urain*, Anqi Li†, Puze Liu*, Carlo D'Eramo*, Jan Peters*‡
*IAS, TU Darmstadt, † University of Washington, ‡MPI for Intelligent Systems
Email: {urain, liu, deramo}@ias.informatik.tu-darmstadt.de, anqil4@cs.washington.edu, jan.peters@tu-darmstadt.de

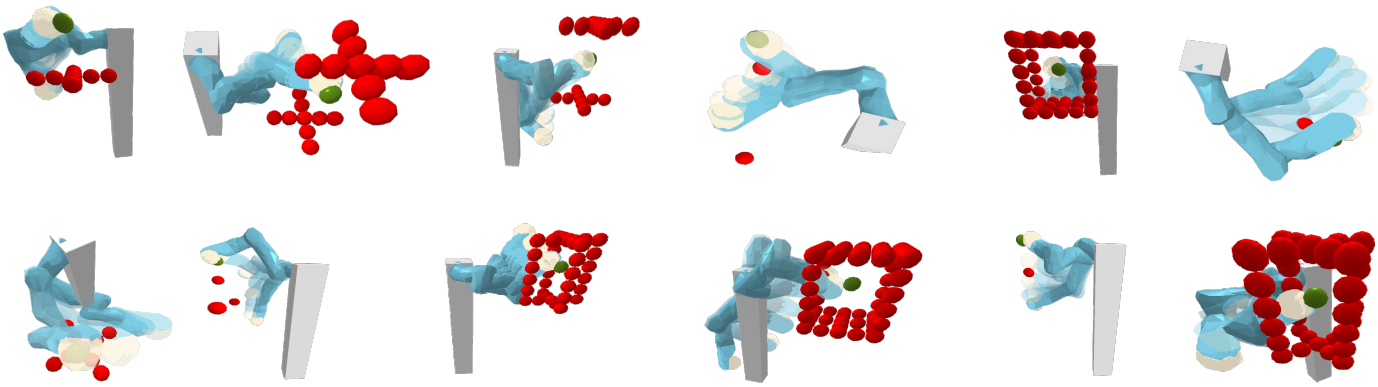


Fig. 1. Composable Energy Policies generate trajectories reactively in a wide set of environments, as the ones in the figure. The generated trajectories are attracted to a target pose, visualized by the green sphere, and avoid a set of obstacles, shown as red spheres.

***Abstract*—Reactive motion generation problems are usually solved by computing actions as a sum of policies. However, these policies are independent of each other and thus, they can have conflicting behaviors when summing their contributions together. We introduce Composable Energy Policies (CEP), a novel framework for modular reactive motion generation. CEP computes the control action by optimization over the product of a set of stochastic policies. This product of policies will provide a high probability to those actions that satisfy all the components and low probability to the others. Optimizing over the product of the policies avoids the detrimental effect of conflicting behaviors between policies choosing an action that satisfies all the objectives. Besides, we show that CEP naturally adapts to the Reinforcement Learning problem allowing us to integrate, in a hierarchical fashion, any distribution as prior, from multimodal distributions to non-smooth distributions and learn a new policy given them. Video in https://sites.google.com/view/composable-energy-policies/home**

I. Introduction

Many robotic tasks deal with finding a control action satisfying multiple objectives. An apparently simple task such as watering some 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 and break plant's branches with its arms. In contrast with more sequential tasks [41, 16, 17, 38], in which the objectives are satisfied concatenating them in time, in the presented work, we consider tasks in which multiple 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 [25, 24, 19] find a global trajectory from start to goal by a computationally intense Monte-Carlo sampling process. Trajectory optimization methods [44, 31, 18, 36] reduce the computational burden of planning methods by learning the global trajectory given an initial trajectory candidate. These methods reshape the global trajectory to satisfy the objectives. However, they still require solving an optimization problem over long temporal horizon trajectories. Reactive Motion Generators, such as Artificial Potential Fields (APF) methods [20, 22, 10, 21, 32, 3] have a very low computational cost, but lack any guarantees of finding a global trajectory satisfying the objectives. These methods propose a modular approach. Each component proposes a deterministic policy to satisfy one of the objectives, and the control action is obtained by the sum of the policy contributions.

There is a clear gap in methodology. The first (path planning and trajectory optimization), approach the problem as an optimization or inference problem over the combined objectives, while the second (reactive motion generation) assumes a complete independence between objectives, lacking any optimality guarantees. We frame Reactive Motion Generation as an optimization problem over a product of expert policies [13, 30, 29, 11, 42]

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \prod_k \pi_k(\boldsymbol{a}|\boldsymbol{s}). \tag{1}$$

Most of the previous approaches for reactive motion generation, compute each action maximizing a certain objective independently and then, sum the actions together [32, 20, 10, 14]. This independence between policies can end up in conflicting behaviors leading to oscillations or local minima. In our

approach, we first compute the product of a set of stochastic policies, and then, we compute the action maximizing the product of them. This approach can be understood as a probabilistic logical conjunction (AND operator) [7, 42] between the stochastic policies (see Fig. 2 for visual representation).

Nevertheless, there are exceptions. Dynamic Window Approach (DWA) [9, 46] 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 computed solving an optimization problem. In contrast with DWA, in our work, we compute the optimal action only through an optimization phase. Additionally, we consider 7-dof robots instead of planar robots and allow to integrate objectives defined in arbitrarily different task spaces. Finally, given the flexibility of our method, we can integrate policies from multiple sources in a single model. We could integrate obstacle avoidance policies with data-driven learned policies, compute a solution by trajectory optimization and add reaction to unexpected situations or integrate priors in a Reinforcement Learning (RL) policy to accelerate learning and ensure safety.

***Contribution**:* We present Composable Energy Policies (CEP), a new framework for modular robot control. In contrast with previous methods [20, 32, 34] that solves reactive motion generation with a sum of deterministic policies, our method solves reactive motion generation by maximum likelihood estimation over a product of expert policies. We claim that optimizing over the composed policy distributions will increase the guarantees of satisfying all the objectives jointly w.r.t. adding actions coming from different policies. To validate our claims, we derive the reactive motion generation problem from control as inference view and compute the optimality guarantees of applying a product of expert policies (Appendix A).

In our work, we aim for flexible composition of modular policy distributions. We propose a framework to integrate stochastic policies represented in arbitrary state-action spaces. Additionally, these policies might have multiple sources from data-driven learned multimodal policies to computationally costly control as inference solutions. In section V, we show how to integrate a RL policy with a set of prior policies to improve exploration and guarantee safety exploration.

In the following, we focus on presenting the method's main blocks while in the Appendix, we focus on the theoretical interpretation of the method. In section VI, we validate our algorithm in both reactive motion generation and prior based RL problems.

***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^x) to represent the space in which the policy is and subscript (π_x) to represent the policy index. $\boldsymbol{f}_x^z(\cdot)$ represent a transformation map from space $\mathcal{X}$ to space $\mathcal{Z}$, $\boldsymbol{s}^x$ is the state in space $\mathcal{X}$ and $\boldsymbol{s}^z$, the state in space $\mathcal{Z}$.*

II. Preliminaries

A. Artificial Potential Fields

Reactive motion generation deals with the problem of generating local and quick motion. The developed methods require to have low computational burden to give a fast response reacting to unexpected situations. APF [20] propose to solve the problem by a weighted sum of a set of dynamic systems represented in a set of task spaces. Let's assume a set of transformations maps $\boldsymbol{f}_x^{z_0}, \dots, \boldsymbol{f}_x^{z_K}$, that transform a point in the configuration space, $\mathcal{X}$, to a set of task spaces, $\mathcal{Z}_k$, $(\boldsymbol{z}_k, \dot{\boldsymbol{z}}_k) = (\boldsymbol{f}_x^{z_k}(\boldsymbol{x}), \boldsymbol{J}^{z_k}(\boldsymbol{x})\dot{\boldsymbol{x}})$, with $\boldsymbol{J}^{z_k} = \frac{\partial \boldsymbol{f}_x^{z_k}(\boldsymbol{x})}{\partial \boldsymbol{x}}$; and a set of deterministic second-order dynamic systems in the task space, $\boldsymbol{g}^{z_0}, \dots, \boldsymbol{g}^{z_k}$. The APF represents the dynamic system in the configuration space $\mathcal{X}$

$$\boldsymbol{g}^x = \sum_{k=0}^{K} \boldsymbol{J}^{z_k^+} \boldsymbol{\Lambda}_k(\boldsymbol{z}_k) \boldsymbol{g}^{z_k} \tag{2}$$

with a weighted ($\boldsymbol{\Lambda}_k(\boldsymbol{z}_k)$) sum of the of the projected dynamics and $\boldsymbol{J}^{z_k^+}$ the pseudoinverse Jacobian. Each dynamic component represent the policy to satisfy a particular objective. Given that we are looking for the motion that satisfy multiple objectives, the sum of the dynamics might not be enough as the different components might have conflicting behaviours and thus lead to local minima or oscillations. In Appendix C, we show that APF can be rewritten as a particular case of product of experts and provide an interpretation of their failures.

We aim to solve the conflicts between dynamics by the maximum likelihood estimation over the product of a set of dynamic distributions. In our work, we model each dynamic component by a distribution. These distributions can be understood as a weighted set of possible dynamics for each component. In the combination of these distributions, our method will search for the dynamics that better satisfy the combined set of objectives and thus, avoid conflicting behaviors, as shown in Figure 2.

III. Composable Energy Policies

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

$$\pi_i(\boldsymbol{a}|\boldsymbol{s}) = \exp(E_i(\boldsymbol{a}, \boldsymbol{s})) \frac{1}{Z_i(\boldsymbol{s})} \tag{3}$$

with $E : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is an arbitrarily represented energy function and $Z(\boldsymbol{s}) = \int_{\boldsymbol{a}} \exp(E(\boldsymbol{a}; \boldsymbol{s}))d\boldsymbol{a}$ is the normalization factor. Choosing a Boltzmann distribution is not an arbitrary choice. Boltzmann distribution allows representing an arbitrary distribution by a suitable definition of the energy function (E) [26]. Additionally, computing the product of experts

$$\pi(\boldsymbol{a}|\boldsymbol{s}) = \prod_{k=0}^{K} \pi_k(\boldsymbol{a}|\boldsymbol{s})^{\beta_k} \propto \exp\left(\sum_k \beta_k E_k(\boldsymbol{a}, \boldsymbol{s})\right). \tag{4}$$

will end up in a weighted sum over the individual energy components in the log-space. Linearly combined energies are beneficial for computing the energies contribution in parallel

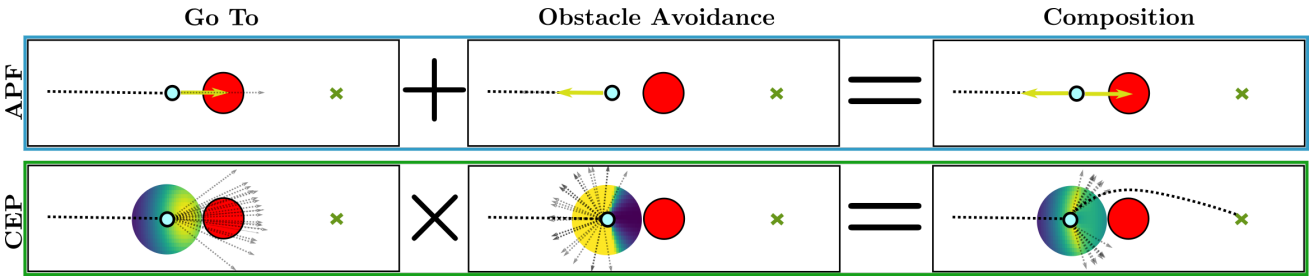


Fig. 2. Visual Representation of modular control for Goto + Obstacle Avoidance. In the top box, we show Artificial Potential Field (APF) [20] policy. In the bottom box, we show Composable Energy Policies (CEP). In contrast, with artificial potential field method, that sums deterministic actions (goto, avoid obstacle), CEP computes the product of the policy distributions. The composition will provide high probability to those actions that satisfy both components and low to the rest. This approach helps avoiding conflicts between different components. *Robot: blue circle, obstacle: red circle and target: green cross. Thick dotted line: performed trajectory, light dotted line: possible future trajectories.*

and thus, in practise, parallelize the computation by multi-processing, increasing the control frequency. Product of experts is a natural choice for our policy; given a set of energies $E_1, \dots, E_K$, product of experts is the posterior of an inference problem maximizing the energies (See Appendix B).

A. Energy Tree

In the composition proposed in (4), 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. Navigation of the robot's end-effector towards a target while avoiding the obstacles, composes skills defined in different task spaces. Each robot link should avoid obstacles and the end effector should reach a certain target position. Inspired by APF [20] and Riemannian Motion Policies (RMP) [32], we propose to model the composition of energies in different task spaces. We introduce the probabilistic graphical model for our policy in Fig.3.

Our architecture is composed of two main components. First, we have a set of policies $\pi^z(\boldsymbol{a}^z|\boldsymbol{s}^z)$, defined in different state-action latent spaces (task space) $(\boldsymbol{s}^z, \boldsymbol{a}^z) \in \mathcal{Z}$. Second, we consider a set of deterministic mappings that transform the state-actions in the common space (configuration space) $(\boldsymbol{s}^x, \boldsymbol{a}^x) \in \mathcal{X}$ to the latent state-action spaces $\mathcal{Z}$, $\boldsymbol{f}_x^z : \mathcal{X} \to \mathcal{Z}$.

Applying the change of variable rule in probabilistic density functions [1, 2], given the mapping $\boldsymbol{f}_x^z$ is bijective in the actions ($\boldsymbol{a}^z \leftrightarrow \boldsymbol{a}^x$), we can write the policy in the common space, π^x, in terms of the latent space policy π^z [45]

$$\begin{aligned}\pi^x(\boldsymbol{a}^x|\boldsymbol{s}^x) &= \pi^z(\boldsymbol{a}^z|\boldsymbol{s}^z)\sqrt{\det(\boldsymbol{J}^\intercal\boldsymbol{J})} \\ &= \pi^z(\boldsymbol{f}_x^z(\boldsymbol{s}^x, \boldsymbol{a}^x))\sqrt{\det(\boldsymbol{J}^\intercal\boldsymbol{J})}. \end{aligned} \tag{5}$$

with $\boldsymbol{J} = \partial \boldsymbol{f}_x^z(\boldsymbol{a}^x, \boldsymbol{s}^x)/\partial \boldsymbol{a}^x$ is the Jacobian of $\boldsymbol{f}_x^z$ in the action $\boldsymbol{a}^x$. The equality in (5) does not hold for injective or surjective transformations. In practice, we apply the equality for any possible map $\boldsymbol{f}_x^z$ (bijective, surjective, and injective), and we leave for future work the study of the policy distribution under surjective or injective mappings.

From (5), we can write $\pi^x(\boldsymbol{a}^x|\boldsymbol{s}^x)$ w.r.t. the energy function defined in the latent space $E^z(\boldsymbol{a}^z; \boldsymbol{s}^z)$

$$\pi^x(\boldsymbol{a}^x|\boldsymbol{s}^x) = \exp\left(E^z(\boldsymbol{f}_x^z(\boldsymbol{a}^x, \boldsymbol{s}^x))\right)\frac{1}{Z(\boldsymbol{s}^x)} \tag{6}$$

with

$$Z = \frac{\int_{\boldsymbol{a}^z}\exp(E^z(\boldsymbol{a}^z, \boldsymbol{s}^z))d\boldsymbol{a}^z}{\sqrt{\det(\boldsymbol{J}^\intercal\boldsymbol{J})}}. \tag{7}$$

This policy representation allows to define the energy function in an easy to represent task space. Then, for any $\boldsymbol{s}^x, \boldsymbol{a}^x \in \mathcal{X}$, we can map them to the latent space $\mathcal{Z}$, $(\boldsymbol{s}^z, \boldsymbol{a}^z) = f_x^z(\boldsymbol{s}^x, \boldsymbol{a}^x)$ and compute the un-normalized log-probability of the action in a given latent policy $E^z(\boldsymbol{a}^z, \boldsymbol{s}^z)$.

Our energy tree is inspired by the APF formulation, but it has important differences. In APF formulation, there is an explicit function from state $\boldsymbol{s}^x$ to action $\boldsymbol{a}^x$

$$\boldsymbol{a}^x = \boldsymbol{f}_x^{z-1}(\boldsymbol{a}^z) = \boldsymbol{f}_x^{z-1}(\boldsymbol{g}(\boldsymbol{s}^z)) = \boldsymbol{f}_x^{z-1}(\boldsymbol{g}(\boldsymbol{f}_x^z(\boldsymbol{s}^x))) \tag{8}$$

where $\boldsymbol{f}_x^z$ is the function mapping from common space to latent space and $\boldsymbol{g}$ is a deterministic policy in the latent space. In this approach, $\boldsymbol{f}_x^z$ is required to be invertible. In contrast, CEP assumes an implicit function from $\boldsymbol{s}^x$ to $\boldsymbol{a}^x$

$$\boldsymbol{a}^x = \arg\max_{\boldsymbol{a}^x} E^z(\boldsymbol{f}_x^z(\boldsymbol{a}^x, \boldsymbol{s}^x)). \tag{9}$$

In this approach, we do not need to compute the inverse of $\boldsymbol{f}_x^z$. Additionally, E^z represents the energy in the latent space, so we can combine distributions instead of combining deterministic dynamics. In Sec. IV we show that having an implicit function might be beneficial for reactive motion generation.

In case a set of policies exist in different task spaces $\mathcal{Z}_1, \dots, \mathcal{Z}_N$, we can compose the policies together

$$\pi^x(\boldsymbol{a}^x|\boldsymbol{s}^x) = \exp\left(\sum_{k=1}^K \beta_k E^{z_k}(\boldsymbol{f}_x^{z_k}(\boldsymbol{a}^x, \boldsymbol{s}^x))\right)\frac{1}{Z(\boldsymbol{s}^x)} \tag{10}$$

Shown in Fig. 3, CEP has a recursive architecture. Our model allows us to represent the latent policies $\pi^z(\boldsymbol{a}^z|\boldsymbol{s}^z)$ with a set of policies in an even deeper latent space $\mathcal{W}$, and still compute the optimization in the same way.

B. Maximum Likelihood Estimation on CEP

In CEP, the optimal action is obtained by Maximum Likelihood Estimation over the product of expert policies. Given the current state $\boldsymbol{s}^x$

$$\boldsymbol{a}^x_{\text{ML}} = \arg\max_{\boldsymbol{a}^x} \prod_{k=0}^K \pi_k(\boldsymbol{a}^x|\boldsymbol{s}^x)^{\beta_k}\,,\ \beta_k > 0. \tag{11}$$

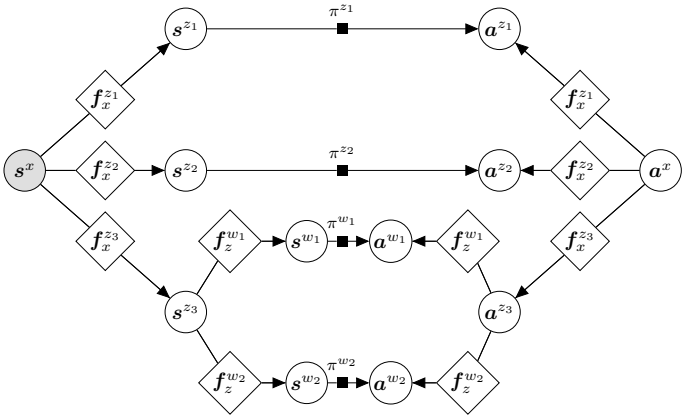


Fig. 3. Bayesian network for the energy tree. The policy distribution $\pi^x(\boldsymbol{a}^x|\boldsymbol{s}^x)$ is computed w.r.t. the policies in the latent spaces $\mathcal{Z}_i$. Additionally, a latent policy $\pi^z(\boldsymbol{a}^z|\boldsymbol{s}^z)$ can be represented by a set of policies in a different latent spaces $\mathcal{W}_i$. *White diamonds represent deterministic transformations and black square is a probabilistic mapping.*

We assume a set of mappings $\boldsymbol{f}_x^z$ and a set of energies E^z are given. We solve (11), by Cross Entropy optimization [35]. We define a proposed sampling model $p(\boldsymbol{a}|\boldsymbol{\theta})$ and iteratively solve

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \arg\max_{\boldsymbol{\theta}_t} \mathbb{E}_{p(\boldsymbol{a}^x|\boldsymbol{\theta}_t)}\left[\log\left(\prod_{k=0}^{K}\pi_k(\boldsymbol{a}^x|\boldsymbol{s}^x)^{\beta_k}\right)\right] \\ &\propto \mathbb{E}_{p(\boldsymbol{a}^x|\boldsymbol{\theta}_t)}\left[\sum_{k=1}^{K}\beta_k E^{z_k}(\boldsymbol{f}_x^{z_k}(\boldsymbol{a}^x,\boldsymbol{s}^x))\right]. \end{aligned} \tag{12}$$

The method is presented in Algorithm 1. Each energy (E^z) component is linearly dependent and thus, in the optimization process, we evaluate each energy component in parallel. Also, we consider a batch N of actions evaluated all in parallel on GPU. This parallelization reduces the computational time for the optimization process and allows to use CEP for high-frequency reactive motion generation.

Algorithm 1: Composable Energy Policies

Given: N: Number of samples;
$\boldsymbol{s}^x$: Current state in common space;
K: Number of Energy Components;
$(\boldsymbol{f}_x^{z_1}, E^{z_1}, \beta^{z_1}), \ldots, (\boldsymbol{f}_x^{z_K}, E^{z_K}, \beta^{z_K}))$: Maps, energies and inverse temperatures;
I: Optimization steps;
(μ_0, Σ_0): Initial sampling distribution mean and variance;
$(\boldsymbol{a}^*, e^*)$: Initial optimal action and energy;

for $i \leftarrow 0$ **to** $I-1$ **do**
  for $n \leftarrow 0$ **to** $N-1$ **do**
    $\boldsymbol{a}_n^x \sim \mathcal{N}(\mu_i, \Sigma_i)$;
    for $k \leftarrow 1$ **to** K **do**
      $\boldsymbol{s}^{z_k}, \boldsymbol{a}_n^{z_k} = \boldsymbol{f}_x^{z_k}(\boldsymbol{s}^x, \boldsymbol{a}_n^x)$;
      $e_n^{z_k} = E^{z_k}(\boldsymbol{s}^{z_k}, \boldsymbol{a}_n^{z_k})$;
    $e_n^x = \sum_{k=1}^{K}\beta^{z_k} e_n^{z_k}$
  $\mu_{i+1} \leftarrow \text{Update}_\mu(\mu_i, \boldsymbol{a}_{0:N}^x, e_{0:N}^x)$;
  $\Sigma_{i+1} \leftarrow \text{Update}_\Sigma(\Sigma_i, \boldsymbol{a}_{0:N}^x, e_{0:N}^x)$;
  $\boldsymbol{a}_i^*, e_i^* \leftarrow \arg_{\boldsymbol{a}}\max_e(e_{0:N}^x)$;
  if $e^* < e_i^*$ **then**
    $\boldsymbol{a}^* \leftarrow \boldsymbol{a}_i^*$;
    $e^* \leftarrow e_i^*$;
return $\boldsymbol{a}^*$;

IV. CEP for Robot Motion Generation

In this section, we will provide further insights on how to model the mapping functions and the stochastic policies for a robot motion generation problem. We consider the motion is generated with a second-order dynamic system in the robot's configuration space

$$\ddot{\boldsymbol{q}} = g(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{13}$$

where $\boldsymbol{q}$ is the robot's joint state, $\dot{\boldsymbol{q}} = d\boldsymbol{q}/dt$ velocity and $\ddot{\boldsymbol{q}} = d^2\boldsymbol{q}/dt^2$, the acceleration.

Our dynamic system is modeled by the maximization in (11). Thus, we represent our action $\boldsymbol{a}^x = \ddot{\boldsymbol{q}}$ and the state $\boldsymbol{s}^x = \{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$.

The energy distributions are defined in a set of task spaces. We model the map between the common space (configuration space) and the latent space (task space) $\boldsymbol{f}_q^x$ by the robot's kinematics

$$\begin{aligned}\boldsymbol{x} &= \mathbf{f}_{\text{kin}}(\boldsymbol{q}) \\ \dot{\boldsymbol{x}} &= \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{x}} &= \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} \approx \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}}\end{aligned} \tag{14}$$

with forward kinematics $\mathbf{f}_{\text{kin}}$ to a given task space and $\boldsymbol{J}(\boldsymbol{q}) = \partial\boldsymbol{x}/\partial\boldsymbol{q}$, the Jacobian for the given forward kinematics.

For the particular case in which the mappings are given by (14), we represent the general policy in (10) as

$$\pi^q(\ddot{\boldsymbol{q}}|\boldsymbol{q}, \dot{\boldsymbol{q}}) \propto \exp\left(\sum_{k=1}^{K}\beta_k E^{x_k}(\boldsymbol{J}_k(\boldsymbol{q})\ddot{\boldsymbol{q}}, \mathbf{f}_{\text{kin}k}(\boldsymbol{q}), \boldsymbol{J}_k(\boldsymbol{q})\dot{\boldsymbol{q}})\right). \tag{15}$$

We remark, that computing the optimal solution for (15), given we use an implicit representation, we only require access to the forward kinematics $\mathbf{f}_{\text{kin}}$ and the Jacobian $\boldsymbol{J}$ functions. Instead, explicit methods [32, 20], require to compute the Jacobian pseudoinverse to get the action in the configuration space as shown before in (2) and (8). Jacobian pseudoinverse might be problematic to compute if the robot is in singularities.

Energy functions allow having a flexible policy distribution representation. In the following, we provide some energy proposals for the main components the robot requires for reactive motion generation. Additionally, we show in Appendix C, that

the weighted sum of deterministic policies (Artificial Potential Fields methods), can be framed as a particular case of CEP.

Go To a target: The simplest distribution model for going to a target position is a normal distribution

$$\pi(\ddot{\boldsymbol{x}}|\boldsymbol{x},\dot{\boldsymbol{x}}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x},\dot{\boldsymbol{x}}),\boldsymbol{\Sigma}(\boldsymbol{x})) \tag{16}$$

with

$$\boldsymbol{\mu}(\boldsymbol{x},\dot{\boldsymbol{x}}) = -\boldsymbol{K}_p(\boldsymbol{x}-\boldsymbol{x}^*) - \boldsymbol{K}_v\dot{\boldsymbol{x}} \tag{17}$$

$$\boldsymbol{\Sigma}(\boldsymbol{x}) = \alpha(\boldsymbol{x}-\boldsymbol{x}^*)^\intercal(\boldsymbol{x}-\boldsymbol{x}^*). \tag{18}$$

The distribution proposes as mean $\boldsymbol{\mu}$, a PD-controller with $\boldsymbol{x}^*$ the target state and $\boldsymbol{K}_p > 0$ and $\boldsymbol{K}_v > 0$ gains and a covariance matrix $\boldsymbol{\Sigma}$ represented as the quadratic distance to the target, scaled by $\alpha > 0$. The variance will shrink closer to the target and, thus, the relevance of this energy component will increase the closer we are to the desired state.

Obstacle Avoidance: We represent obstacle avoidance energy in the unidimensional space represented by the vector between the cartesian robot position $\boldsymbol{x}_r$ in task space and the cartesian obstacle position $\boldsymbol{x}_o$.

We first compute the vector pointing to the obstacle

$$\boldsymbol{v}_{r,o} = \boldsymbol{x}_r - \boldsymbol{x}_o \tag{19}$$

Then, we project the velocity $\dot{\boldsymbol{x}}$ and the acceleration $\ddot{\boldsymbol{x}}$ vectors in the task space

$$\begin{aligned}\dot{x}_p &= \dot{\boldsymbol{x}}\frac{\dot{\boldsymbol{x}}\cdot\boldsymbol{v}_{r,o}}{||\dot{\boldsymbol{x}}||||\boldsymbol{v}_{r,o}||}\\ \ddot{x}_p &= \ddot{\boldsymbol{x}}\frac{\ddot{\boldsymbol{x}}\cdot\boldsymbol{v}_{r,o}}{||\ddot{\boldsymbol{x}}||||\boldsymbol{v}_{r,o}||}\end{aligned} \tag{20}$$

Finally, we compute the energy for the acceleration $\ddot{x}_p$, $\dot{x}_p$ and $\boldsymbol{v}_{r,o}$ as

$$E(\ddot{x}_p,\dot{x}_p,\boldsymbol{v}_{r,o}) = \begin{cases}0 & \text{if } \dot{x}_p < 0 \text{ or } ||\boldsymbol{v}_{r,o}|| > \gamma\\ -\infty & \text{if } \ddot{x}_p > -\alpha\dot{x}_p - \beta\\ 0 & \text{otherwise}\end{cases} \tag{21}$$

with a distance threshold parameter $\gamma > 0$ to activate the energy component. If the projected robot velocity is not pointing to the obstacle or if the robot's task space position is too far from the robot, the distribution is a uniform distribution over the whole acceleration space. In case the robot is close to the obstacle and the velocity vector points to the obstacle, the distribution then becomes a uniform distribution between $-\infty$ and $-\alpha\dot{x}_p - \beta$, $\mathcal{U}_p(-\infty, -\alpha\dot{x}_p - \beta)$ with $\alpha > 0$ and $\beta > 0$.

Avoid Joint Limits: Similarly to the collision avoidance energy, we model the joint limits avoidance with a uniform distribution that provides high probability to those accelerations pointing in the opposite direction to the joint limit as long as the joint is close to the limits. Given the distance to the joint limit

$$d_l = q - q_l \tag{22}$$

with q_l the joint limit, the energy can be represented as

$$E(\ddot{q};\dot{q},q) = \begin{cases}0 & \text{if } \frac{d_l}{|d_l|}\dot{q} < 0 \text{ or } |d_l| > \gamma\\ -\infty & \text{if } \frac{d_l}{|d_l|}\ddot{q} > -\alpha\frac{d_l}{|d_l|}\dot{q} - \beta\\ 0 & \text{otherwise}\end{cases} \tag{23}$$

with $\alpha > 0$ and $\beta > 0$.

Imitation Learning, Control-as-Inference and RL: CEP architecture is not only limited to heuristic energies. The architecture allows to integrate any form distributions. CEP architecture allows to integrate multimodal policies learned by behavioural cloning [45, 29], posterior distributions computed by control as inference [23, 48, 49] or a RL policy.

In the following, we propose a novel hierarchical reinforcement learning policy to integrate an RL agent with a set of prior distributions.

V. CEP for Hierarchical Reinforcement Learning

Priors have been widely used in RL, accelerating the learning process and therefore improving the sample efficiency. Most popular approaches can be categorized in two groups. On the one hand, methods that first approximate a parameterized policy to mimic the priors

$$\min_\theta D_{\mathrm{KL}}(\pi_\theta(\boldsymbol{a}|\boldsymbol{s})||\pi_q(\boldsymbol{a}|\boldsymbol{s}))$$

and then, update this policy to maximize for a new task [28, 45].

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

Even if this approach is the most popular one, if the parameterized policy is not sufficiently expressive is not going to cover all the prior information. Moreover, the parameterized model might forget important information coming from the prior after some RL updates. On the other hand, Hierarchical Reinforcement Learning (HRL) methods [15, 39, 40, 5, 30] consider a two-layered policy.

$$\pi^L(\boldsymbol{a}|\boldsymbol{s}) = \int_{\boldsymbol{a}_H} \pi_q(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{a}_H)\pi^H_\theta(\boldsymbol{a}_H|\boldsymbol{s})d\boldsymbol{a}_H \tag{24}$$

Given some priors, a policy π_q is modeled conditioned on a higher-order action $\boldsymbol{a}_H$. For sampling an action, we first sample an action from the high-level policy, $\boldsymbol{a}_H \sim \pi^H_\theta(\boldsymbol{a}_H|\boldsymbol{s})$ and then, we sample from the conditioned low-level policy, $\boldsymbol{a} \sim \pi_q(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{a}_H)$. Most of the previous HRL policies assume rather a mixture of skills [30, 12, 28, 8] or a simple sum of the prior actions [15, 39] as HRL policies. These policy architectures might not satisfy the desired properties in a real robot RL problem, rather because they are not safe enough or because they are so limited in their exploration space.

In a robot learning scenario, a desired HRL policy should consider (1) priors that guarantee hard constraints to avoid collisions, (2) multimodal/non-normal priors to provide a rich set of possible exploring regions, (3) allow to integrate multiple prior sources in the same policy and (4) do not limit the exploration region to a few set of given skills. In the

following, we show that CEP allows integrating all the desired properties in the HRL policy.

In our work, we propose to model the low level policy with a CEP

$$\boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H \sim \pi_\theta^H(\boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H|\boldsymbol{s}) \tag{25}$$
$$\pi^L(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H) = \pi_q(\boldsymbol{a}|\boldsymbol{s})\mathcal{N}(\boldsymbol{a}|\boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H)$$

In our model, the action is computed by the maximization over the low-level policy

$$\boldsymbol{a} = \arg\max_{\boldsymbol{a}} \pi^L(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H). \tag{26}$$

Our proposed policy combines the given prior policy distribution with a normal distribution parameterized with the high-level policy. This structure allows integrating any type of prior distribution with a learnable policy. The prior could impose hard constraints to avoid actions that lead to collisions by setting $\pi_q(\boldsymbol{a}|\boldsymbol{s}) = 0$ or encourage the robot to explore in a mixture of regions by setting a mixture of distributions as prior.

The variance $\boldsymbol{\Sigma}_H$ control controls the importance of the learning policy w.r.t. the priors. The bigger $\boldsymbol{\Sigma}_H$ is the less relevant the learning policy is and thus, the higher the influence of the priors in the action; in the limit $\lim_{\boldsymbol{\Sigma}_H \to 0} \pi^L(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H) = \delta(\boldsymbol{\mu}_H - \boldsymbol{a})$, the prior effect disappear and we only trust in the learning policy.

It is interesting to remark that our model applies RL in the high-level policy and solves a stochastic search optimization problem in the low-level policy. With the low-level stochastic search optimization, we can guarantee that some conditions (obstacle avoidance) will be satisfied in the robot action, while, a simple residual control could not.

VI. Experimental Evaluation

The model evaluation is split into two parts. First, we study CEP performance for reactive motion generation. We evaluate how the different energy components perform by observing the success rate and the collisions in a set of obstacle avoidance environments. We compare CEP w.r.t. previous modular reactive controllers. We exclude trajectory optimization and path planning algorithms as our evaluation is interested in reactive controllers that can provide a control response in high control frequencies ($\geq$500Hz).

Second, we evaluate CEP in an RL problem. We want to observe if using CEP as prior boosts the learning performance of an RL agent. Additionally, we want to evaluate if the proposed priors are sufficiently strong to impose safety priors in the policy and avoid collisions. We compare our approach to previous methods applying both *Behavioural Cloning + RL* and *Hierarchical RL*.

Reactive Motion Generation

We consider the problem of controlling a robot manipulator in a set of obstacle avoidance environments. We use a 7 dof KUKA-LWR robot manipulator and the objective is to reach a target pose (3D and 6D) while avoiding colliding with the obstacles. We consider increasingly difficult environments presented in Fig. 4. The robot is initialized in a random joint configuration and its motion is generated by a set of modular reactive motion generators, without additional global path planning algorithms. We evaluate the success rate and the collision rate of our method and compare it w.r.t. deterministic modular reactive motion controllers (Artificial Potential Fields [20] and Riemannian Motion Policies [32]) as baselines.

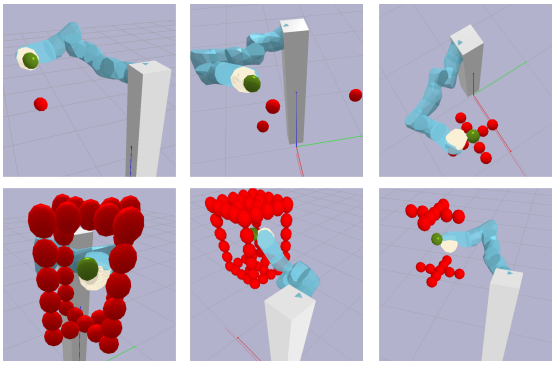


Fig. 4. Reactive Motion Generation Environments. In the top, from left to right: 1 Obstacle, 3 Obstacles, Cross. In the bottom, from left to right: Cage I, Cage II, Double Cross. *Green Sphere: target pose, red spheres: obstacles.*

We build the CEP model with

- Go-To energy in the end effector's task space.
- Obstacle Avoidance energy in each robot link's task space.
- Joint limits avoidance energy in the configuration space.

To guarantee high frequency control actions ($\geq 500Hz$), we solve the optimization problem (11) in a Nvidia GPU RTX-2800. Each energy component is parallelized and the action samples are evaluated in batch. With a sufficient amount of samples per optimization step (10.000 samples), the optimization is solved in two steps.

***Results and analysis**:* We summarize the obtained results for the 3D Goto problem in table I. Few obstacles environments are easily solved by all the methods. We observe a success rate of almost 100% for the first three environments in the three 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 an 89% success rate in the first cage environment and 46% in the second cage. We think that these results might be related to the way we compute the robot's acceleration. While the baselines methods compute the accelerations provided by the different components(Go-To, Avoid Joint Limits, Obstacle Avoidance), and do not take into account any possible conflict of interests between each action, our method first computes the distribution obtained by the product of all the components and then, computes the action that maximizes this composed distribution. The built composition will provide high probabilities to those actions that have high probability in each component and low to the rest. Thus, the optimal action from

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 [32]	100/100	0/100	98/100	0/100	94/100	0/100	84/100	0/100	55/100	0/100	5/100	0/100
Artificial Potential Fields [20]	100/100	0/100	100/100	0/100	90/100	0/100	76/100	0/100	43/100	0/100	4/100	2/100
Composable Energy Policies(Ours)	**100/100**	**0/100**	**100/100**	**0/100**	**98/100**	**0/100**	**91/100**	**0/100**	**89/100**	**0/100**	**46/100**	**0/100**

TABLE I
RESULTS FOR 3D GOTO + OBSTACLE AVOIDANCE TASK

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 [32]	100/100	0/100	91/100	0/100	75/100	0/100	63/100	0/100	18/100	0/100	1/100	0/100
Artificial Potential Fields [20]	100/100	0/100	90/100	0/100	78/100	0/100	60/100	0/100	21/100	0/100	1/100	0/100
Composable Energy Policies(Ours)	**100/100**	**0/100**	**100/100**	**0/100**	**91/100**	**0/100**	**87/100**	**0/100**	**43/100**	**0/100**	**14/100**	**0/100**

TABLE II
RESULTS FOR 6D GOTO + OBSTACLE AVOIDANCE TASK

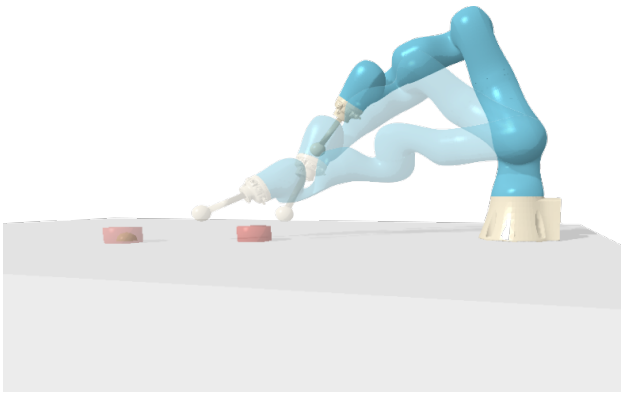


Fig. 5. Puck hitting environment. CEP is tested in a RL environment where the robot needs to learn how to hit the puck. The reward is defined by the euclidean distance between the puck and the green target. An additional reward penalizes if the robot hits the table.

the composed distribution is expected to be the one that better satisfies all the individual components.

The performance worsens for the 6D GoTo problem (Table II). The orientation sets an important constraint in the possible final configurations and reduces the set of trajectories that solves the problem. CEP was 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.

Prior Based Policies for RL

In the following experiment, we evaluate the performance of CEP as a policy that combines prior policies with a new learnable policy to solve a new task. We consider the problem of hitting a puck and place it in a target position Fig. 5. We use a 7 dof LBR-IIWA robot, the action space is defined by the end-effector's task space cartesian velocity $\boldsymbol{a} \in \mathbb{R}^3$ and 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 $\boldsymbol{x}_{\text{target}}$. One episode has 1400 steps. We define the reward as

$$r = -||\boldsymbol{x}_{\text{puck}} - \boldsymbol{x}_{\text{target}}||^2 + r_T(\boldsymbol{x}_{\text{ee}}) \tag{27}$$

with $r_T(\boldsymbol{x}_{\text{ee}}) = -1000$ if the robot's end effector position $\boldsymbol{x}_{\text{ee}}$ collides the table. The first term $-||\boldsymbol{x}_{\text{puck}} - \boldsymbol{x}_{\text{target}}||^2$ will encourage the robot to put the puck close to the target, minimizing the euclidean distance. The second term will encourage the robot to avoid to collide against the table. We compare CEP approach presented in (25) with other approaches that combines prior information with RL. We consider Behavioural Cloning + RL [28] and Residual Policy Learning [15, 39] as baselines.

To guarantee a fair comparison, we apply the same prior policy in the three scenarios. We build a prior policy with two components in a CEP. We consider

- Go-To energy in the robot's end effector's task space.
- Table avoidance in the robot's end effector's task space.

The prior is combined in different forms depending on the applied algorithm:

Behavioural Cloning + RL. We first fit a policy to the prior policy

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{s})} \left[D_{\text{KL}}(\pi_q(\boldsymbol{a}|\boldsymbol{s}) || \pi_\theta(\boldsymbol{a}|\boldsymbol{s})) \right] \tag{28}$$

and then directly sample from the fit policy

$$\boldsymbol{a} \sim \pi_\theta(\boldsymbol{a}|\boldsymbol{s}). \tag{29}$$

Residual Policy Learning. We consider a hierarchical structure. We first sample, from the learnable policy and add the optimal solution from the prior

$$\boldsymbol{a} = \boldsymbol{a}^* + \boldsymbol{a}_H \, , \, \boldsymbol{a}_H \sim \pi_\theta(\boldsymbol{a}_H|\boldsymbol{s}) \tag{30}$$

with

$$\boldsymbol{a}^* = \arg\max_{\boldsymbol{a}} \pi_{\boldsymbol{q}}(\boldsymbol{a}|\boldsymbol{s}). \tag{31}$$

Composable Energy Policies. We also consider a hierarchical structure. In this case, we first sample from the learnable policy

$$\boldsymbol{a}_H \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_H|\boldsymbol{s}) \tag{32}$$

and then, optimize over the composition

$$\boldsymbol{a} = \arg\max_{\boldsymbol{a}} \pi_q(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{a}_H). \tag{33}$$

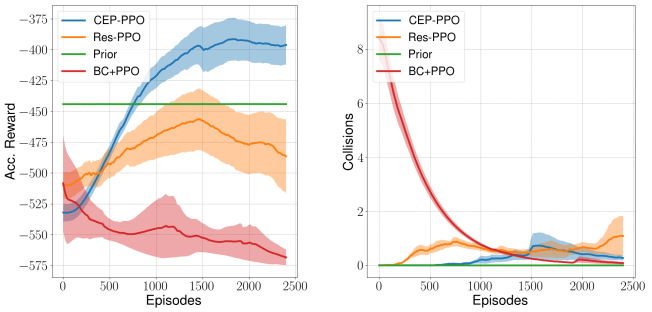


Fig. 6. Training curves for Hitting a puck environment. CEP-PPO performs consistenly better than other prior + RL methods. Collision avoidance prior is stronger in CEP-PPO and provides a safer training.

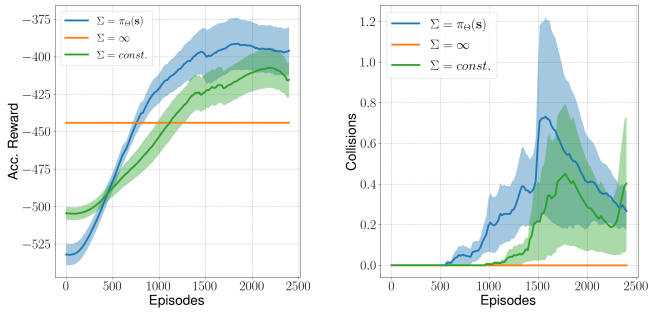


Fig. 7. Training curves for Hitting a puck environment. Learning the variance of the policy improves the learning curve, but also increases the probability of collisions.

We apply the hierarchical policy introduced in (25). For all the experiments, we used the Mushroom-RL [6] PPO [37] implementation. We include additional details in Appendix D.

*a) **Results and analysis**:* We show the obtained results in Fig. 6. Our method, CEP-PPO performs on average better than Residual Learning and Behavioural Cloning + RL methods. The obtained results show that while CEP can consistently improve its performance, residual policy learning has a slower learning curve and behavioral cloning + RL decays in performance. We remark that the three methods start performing worse than the prior due to the stochasticity of the policies in contrast with the deterministic prior. The obtained results might be related to the collisions per episode. CEP based policy allows imposing hard constraints to avoid collisions. Both, behavioral cloning + RL and residual learning adds white noise around the optimal prior action and then, increase the chances of table collision. Fewer collisions mean that $r_T(\boldsymbol{x}_{ee})$ is less activated during the training and then, the learning can focus on moving the puck close to the target. In the Behavioural Cloning case, we see that the table collisions push the robot to avoid the table, and then, it forgets how to reach the target. The Behavioural Cloning + RL policy ends with a policy that makes the robot neither collide with the table and neither to hit the puck. In the residual learning case, the policy is updated by taking both hitting the puck and avoid the table into consideration, while in CEP, the collisions against the table are less, and thus, the training is more focused on improving the hitting tactic.

***CEP ablation study**:* When applying CEP-PPO, the $\boldsymbol{\Sigma}_H$ parameter from (25) controls how big is the influence of the learnable policy in the overall policy. The smaller $\boldsymbol{\Sigma}_H$ is; the smaller the entropy for the learnable part and the higher the influence in the overall policy. In order to study the influence of the $\boldsymbol{\Sigma}_H$ parameter, we compare CEP-PPO with fixed $\boldsymbol{\Sigma}_H$ and CEP-PPO with learnable $\boldsymbol{\Sigma}_H$. We present the obtained results in Fig. 7 As expected having the freedom to set the $\boldsymbol{\Sigma}_H$ parameter allows the robot to learn a better behavior to hit the puck. If the $\boldsymbol{\Sigma}_H$ parameter is fix and too big, then the learned policy has a low impact on the overall policy and it can not improve a lot. With a very small $\boldsymbol{\Sigma}_H$, the learned policy will have a very high impact in the overall policy and then, it might not get benefited by the priors. When learning $\boldsymbol{\Sigma}_H$ parameter in the high-level policy, the robot decides when to trust more in the priors and when less. This allows the robot to get help from priors only if required, but also to decouple from them to learn a better policy. However, shown in the collisions plot from Fig. 7, reducing the effect of the priors might increase the probability to go into dangerous regions and collide more often against the table. Nevertheless, learning the $\boldsymbol{\Sigma}_H$ parameter seems to provide better results than keeping it fixed.

VII. Conclusions

We have presented Composable Energy Policies, a novel approach for modular robot control. In contrast with the previous method for reactive motion generation, which computes the optimal action for each component independently; our method solves an optimization problem over the sum of components. We have shown that APF policies can be rewritten as a CEP and provide reasoning of why their method might fail. Through an evaluation phase, we have also shown that our method can solve a set of reactive motion generation problems more efficiently than artificial potential field methods.

Our method is flexible and allows us to represent each component in an arbitrary set of state-action spaces. Additionally, CEP allows integrating policies from multiple sources and samples from the product of them. In our work, we show how to integrate a RL agent with a set of priors. The resulting algorithm combines a high-level RL policy with a low-level optimization problem. This low-level optimization allows to guarantee that the policy is safe (if an obstacle avoidance prior is added) or to explore in informative regions (if guiding priors are added). We have compared our method with two methods that also combine prior information with an RL agent. Given the low-level optimization of our method, we can guarantee that some priors such as obstacle avoidance were satisfied and the learning is more stable.

Additionally, we have introduced in the Appendix, the relations between optimal control and reactive motion generation. Setting their connections allows us to compute how the optimal policy for a multi-objective optimal control problem diverges from the product of the optimal policies for each objective. These insights allow us to better understand the policy composition and improve the design of our components.

References

[1] Adi Ben-Israel. The change-of-variables formula using matrix volume. *SIAM Journal on Matrix Analysis and Applications*, 21(1):300–312, 1999.

[2] Adi Ben-Israel. An application of the matrix volume in probability. *Linear Algebra and its Applications*, 321 (1-3):9–25, 2000.

[3] Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. Rmpflow: A computational graph for automatic motion policy generation. In *International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[4] Marco Da Silva, Frédo Durand, and Jovan Popović. Linear bellman combination for control of character animation. *ACM SIGGRAPH*, 2009.

[5] Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281. PMLR, 2012.

[6] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Mushroomrl: Simplifying reinforcement learning research. https://github.com/MushroomRL/mushroom-rl, 2020.

[7] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation and inference with energy based models. In *Conference on Neural Information Processing Systems*, 2020.

[8] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *In International Conference on Learning Representations*, 2017.

[9] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[10] Shuzhi Sam Ge and Yun J Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous robots*, 13(3):207–222, 2002.

[11] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pages 6244–6251. IEEE, 2018.

[12] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[13] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[14] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[15] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *International Conference on Robotics and Automation*, pages 6023–6029. IEEE, 2019.

[16] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *IEEE International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

[17] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10): 1194–1227, 2013.

[18] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[19] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[20] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985. doi: 10.1109/ROBOT.1985.1087247.

[21] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 1987.

[22] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, 1991.

[23] Alexander Lambert, Adam Fishman, Dieter Fox, Byron Boots, and Fabio Ramos. Stein variational model predictive control. *Conference on Robot Learning*, 2020.

[24] Steven M LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[25] Steven M LaValle, James J Kuffner, BR Donald, et al. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 5:293–308, 2001.

[26] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[27] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018.

[28] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

[29] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems*, pages 2616–2624, 2013.

[30] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems 32*, 2019.

[31] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[32] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[33] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.

[34] Elon Rimon and Daniel E Koditschek. Exact robot navigation using artificial potential functions. *Departmental Papers (ESE)*, page 323, 1992.

[35] Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology And Computing In Applied Probability*, 1999.

[36] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[39] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[40] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *International Conference on Learning Representations*, 2021.

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

[42] Geraud Nangue Tasse, Steven James, and Benjamin Rosman. A boolean task algebra for reinforcement learning. *Conference on Neural Information Processing Systems*, 2020.

[43] Emanuel Todorov. Compositionality of optimal control laws. *Advances in neural information processing systems*, 22:1856–1864, 2009.

[44] Marc Toussaint. Robot trajectory optimization using approximate inference. In *international conference on machine learning*, pages 1049–1056, 2009.

[45] Julen Urain, Michele Ginesi, Davide Tateo, and Jan Peters. Imitationflows: Learning deep stable stochastic dynamic systems by normalizing flows. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[46] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[47] Benjamin Van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Composing value functions in reinforcement learning. In *International Conference on Machine Learning*, pages 6401–6409, 2019.

[48] Joe Watson, Hany Abdulsamad, and Jan Peters. Stochastic optimal control as approximate input inference. In *Conference on Robot Learning*, pages 697–716. PMLR, 2020.

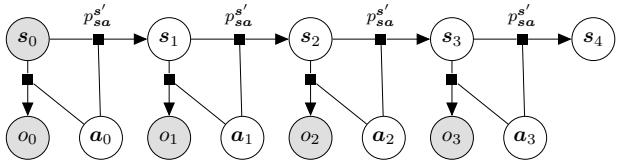
[49] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[50] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning*, 2010.

APPENDIX A
A CONTROL AS INFERENCE VIEW FOR REACTIVE MOTION GENERATION

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

Fig. 8. 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 Bayesian inference problem [33, 27] over the sequence of actions $\boldsymbol{a}_{0:T}$. The optimal control problem is visualized as a graphical model in Fig. 8. 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

$$p(A_0^T|\boldsymbol{s}_0, O_0^T) = \frac{p(O_0^T|A_0^T, \boldsymbol{s}_0)q(A_0^T|\boldsymbol{s}_0)}{p(O_0^T|\boldsymbol{s}_0)} \tag{34}$$

with

$$p(O_0^T|A_0^T, \boldsymbol{s}_0) = \int_{\boldsymbol{s}_{1:T}} p(O_0^T|S_0^T, A_0^T)p(S_1^T|A_0^T, \boldsymbol{s}_0)dS_1^T \tag{35}$$

where $A_0^T : \{\boldsymbol{a}_0, \dots, \boldsymbol{a}_T\}$, $O_0^T : \{o_0, \dots, o_T\}$ and $S_0^T : \{\boldsymbol{s}_0, \dots, \boldsymbol{s}_T\}$. The are two main directions to solve the posterior in (34). 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.

Reactive motion generation instead, solves a one-step-ahead problem. The solutions in reactive motion generation are local rather than global and thus, they are computationally more efficient.

We frame reactive motion generation as a one-step control as inference problem

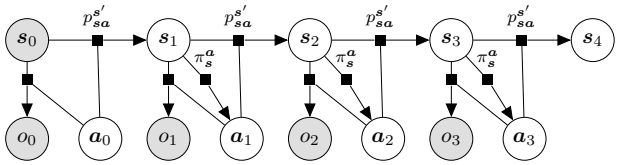
$$p(\boldsymbol{a}_0|\boldsymbol{s}_0, O_0^T) = \frac{p(O_0^T|\boldsymbol{a}_0, \boldsymbol{s}_0)q(\boldsymbol{a}_0|\boldsymbol{s}_0)}{p(O_0^T|\boldsymbol{s}_0)} \tag{36}$$

with

$$p(O_0^T|\boldsymbol{a}_0, \boldsymbol{s}_0) = \int_{S_{1:T}} \int_{A_{1:T}} p(O_0^T|S_0^T, A_0^T)p(S_1^T|\boldsymbol{a}_0, \boldsymbol{s}_0)\pi(A_1^T|S_1^T)dS_1^T dA_1^T. \tag{37}$$

In contrast with the classical stochastic optimal control 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. 9. In

Fig. 9. 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.



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, given we know the policy the agent will apply in the future, we are looking for the instant action the maximized the cumulative reward in the long horizon trajectory. (37) can be rewritten as the expectation over $p(O_0^T|S_0^T, A_0^T)$

$$\mathbb{E}_{S_0^T, A_0^T \sim p^\pi(S_0^T, A_0^T|\boldsymbol{s}_0, \boldsymbol{a}_0)} \left[p(O_0^T|S_0^T, A_0^T)\right] \propto \mathbb{E}_{S_0^T, A_0^T \sim p^\pi(S_0^T, A_0^T|\boldsymbol{s}_0, \boldsymbol{a}_0)} \left[\exp(\sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t))\right] = \exp(Q_r^\pi(\boldsymbol{s}_0, \boldsymbol{a}_0)) \tag{38}$$

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

$$p(\boldsymbol{a}_0|\boldsymbol{s}_0, O_0^T) \propto \exp(Q_r^\pi(\boldsymbol{s}_0, \boldsymbol{a}_0))q(\boldsymbol{a}_0|\boldsymbol{s}_0). \tag{39}$$

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 reactive motion generator can find the optimal trajectory, even if the optimization is done locally.

A. 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 [27] is shown, that the optimal Q function for the control as inference problem can be computed by recursively solving the soft-value iteration [50]

$$\begin{aligned} Q(\boldsymbol{s}, \boldsymbol{a}) &= r(\boldsymbol{s}, \boldsymbol{a}) + \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{s})}\left[V(\boldsymbol{s}')\right] \\ V(\boldsymbol{s}) &= \log \int_{\mathcal{A}} \exp(Q(\boldsymbol{a}, \boldsymbol{s})). \end{aligned} \tag{40}$$

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

For $t = T$,

$$Q^{*T} = \frac{1}{2}(r_1 + r_2) \tag{41}$$

and

$$Q_1^{*T} = r_1\,,\; Q_2^{*T} = r_2. \tag{42}$$

Then,

$$Q^{*T} = \frac{1}{2}(Q_1^{*T} + Q_2^{*T}) \tag{43}$$

and the divergence

$$\Delta Q^T = Q^{*T} - Q_\Sigma^T = 0. \tag{44}$$

From (43), for $T = 1$ horizon optimal control problems, the sum of the optimal components Q_Σ is equal to the optimal Q. For longer horizon control as inference problems, the optimal Q is computed by recursively solving the soft-bellman update backward in time. For computing $t = T - 1$,

$$\begin{aligned} Q^{*T-1}(\boldsymbol{s}, \boldsymbol{a}) &= r(\boldsymbol{s}, \boldsymbol{a}) + \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{s})}\left[V^{*T}(\boldsymbol{s}')\right] \\ V^{*T}(\boldsymbol{s}) &= \log \int_{\mathcal{A}} \exp(Q^{*T}(\boldsymbol{a}, \boldsymbol{s})). \end{aligned} \tag{45}$$

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

$$\Delta Q^{T-1} = Q^{*T-1} - Q_\Sigma^{T-1} = \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}\left[V^{*T} - V_\Sigma^T\right] \tag{46}$$

with

$$\begin{aligned} V^{*T} - V_\Sigma^T &= \log \frac{\int_{\mathcal{A}} \exp(Q^{*T})}{\int_{\mathcal{A}} (\exp(Q_1^{*T}) \int_{\mathcal{A}} \exp(Q_2^{*T}))^{\frac{1}{2}}} \\ &= \log \frac{\int_{\mathcal{A}} \exp(\frac{1}{2}Q_1^{*T}) \exp(\frac{1}{2}Q_2^{*T})}{(\int_{\mathcal{A}} \exp(Q_1^{*T}) \int_{\mathcal{A}} \exp(Q_2^{*T}))^{\frac{1}{2}}}. \end{aligned} \tag{47}$$

Then,

$$\begin{aligned} Q^{*T-1} &= Q_\Sigma^{T-1} + \Delta Q^{T-1} \\ &= Q_\Sigma^{T-1} + \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}\left[\log \frac{\int_{\mathcal{A}} \exp(\frac{1}{2}Q_1^{*T}) \exp(\frac{1}{2}Q_2^{*T})}{(\int_{\mathcal{A}} \exp(Q_1^{*T}) \int_{\mathcal{A}} \exp(Q_2^{*T}))^{\frac{1}{2}}}\right] \end{aligned} \tag{48}$$

From (46) and (47), we can obtain the recurrence relation for the divergence error

$$\Delta Q^{t-1} = \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}\left[\log \frac{\int_{\mathcal{A}} \exp(\frac{1}{2}Q_1^{*t}) \exp(\frac{1}{2}Q_2^{*t}) \exp(\Delta Q^t)}{(\int_{\mathcal{A}} \exp(Q_1^{*t}) \int_{\mathcal{A}} \exp(Q_2^{*t}))^{\frac{1}{2}}}\right]. \tag{49}$$

From (46) and (47), we can see that if $Q_1^* = Q_2^*$, then $\Delta Q = 0$ and the more they differ, the bigger the divergence error to the optima Q^*.

We can also extrapolate interesting results from (49). For any Q_1^* and Q_2^*,

$$(\int_{\mathcal{A}} \exp(Q_1^{*t}) \int_{\mathcal{A}} \exp(Q_2^{*t}))^{\frac{1}{2}} > \int_{\mathcal{A}} \exp(\frac{1}{2}Q_1^{*t}) \exp(\frac{1}{2}Q_2^{*t}) \tag{50}$$

and then,

$$\Delta Q < 0, \forall\, Q_1^*, Q_2^* \tag{51}$$

And then, from (49), ΔQ monotonically decreases. The longer the horizon T, the bigger the divergence error ΔQ.

In practice, these insights are beneficial for modeling our Q functions. We can expect the performance of our reactive motion generation to decay in those environments in which long-horizon planning is required as the ΔQ will increase over long horizons. On the other hand, the more overlapping regions the Q components have, the smaller the divergence error would be and the closer we would be to the optima.

Similar theoretical studies have been already developed [11, 47, 43]. In Optimal Control, composable optimality guarantees were proven for linear dynamics, by the linearly-solvable Markov Decision Processes (LMDP) approach. In [43, 4], 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 [11, 42], the optimality of the composition is studied in a maximum entropy reinforcement learning problem

$$J(\pi) = \mathbb{E}_{\pi, p(\boldsymbol{s}_0)}\left[\sum_{t=0}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha \mathcal{H}(\pi) | \boldsymbol{s}_{t+1} \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \tag{52}$$

with $\mathcal{H}(\pi)$ the entropy of the policy. In [11] is proven, that optimal soft-Q function for $r = \frac{1}{2}r_1 + r_2$ is bounded

$$Q_\Sigma(\boldsymbol{s}, \boldsymbol{a}) \geq Q^*(\boldsymbol{s}, \boldsymbol{a}) \geq Q_\Sigma(\boldsymbol{s}, \boldsymbol{a}) - \mathcal{C}^*(\boldsymbol{s}, \boldsymbol{a}) \tag{53}$$

with $Q_\Sigma = \frac{1}{2}Q_1^* + Q_2^*$ and $\mathcal{C}^*$ is the fixed point of

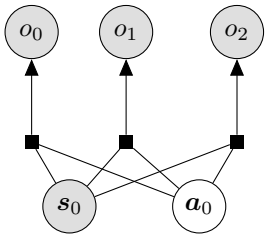
$$\mathcal{C}(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \gamma \mathbb{E}_{p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})}\left[\mathcal{D}_{\frac{1}{2}}(\pi_1^*(\boldsymbol{a}|\boldsymbol{s})||\pi_2^*(\boldsymbol{a}|\boldsymbol{s})) + \max_{\boldsymbol{a}'} \mathcal{C}(\boldsymbol{s}', \boldsymbol{a}')\right] \tag{54}$$

and $\mathcal{D}_{\frac{1}{2}}$ is the Renyi divergence of order $\frac{1}{2}$.

APPENDIX B
COMPOSABLE ENERGY POLICIES AS INFERENCE

The product of experts is the natural expression for the multi-objective inference problem presented in Fig. 10. We introduce a set of auxiliary variables $o_0, \dots, o_K$ that represents the optimality of $\boldsymbol{s}_0$ and $\boldsymbol{a}_0$ given a certain energy function $p(o_k = 1|\boldsymbol{s}_0, \boldsymbol{a}_0) \propto \exp(E_k(\boldsymbol{s}, \boldsymbol{a}))$. Then, the likelihood for

Fig. 10. 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 energy.



the graphical model in Fig. 10 can be computed as the product of the terms

$$p(\boldsymbol{s}_0, \boldsymbol{a}_0, o_{0:2}) = q(\boldsymbol{a}_0)p(\boldsymbol{s}_0)\prod_{k=0}^{2} p(o_k|\boldsymbol{s}_0, \boldsymbol{a}_0). \tag{55}$$

We are interested on computing the maximum a posteriori for $\boldsymbol{a}_0$ given $\boldsymbol{s}_0$ and $o_k = 1\,, \forall\, k = (0, 1, 2)$

$$\begin{aligned}\boldsymbol{a}_0^* &= \arg\max_{\boldsymbol{a}_0} p(\boldsymbol{a}_0|\boldsymbol{s}_0, o_{0:2} = 1) \\ &\propto \exp\left(\sum_{k=0}^{2} E_k(\boldsymbol{a}_0, \boldsymbol{s}_0)\right) q(\boldsymbol{a}_0).\end{aligned} \tag{56}$$

From (56), we can see that the maximum a posteriori will search for the optimal $\boldsymbol{a}_0$ given an exponentiated sum of energies and the prior distribution $q(\boldsymbol{a}_0)$. If we assume a uniform distribution for the prior, we recover (4). Thus, the product of experts represents the distribution that satisfies best a set of components $E_0, \dots, E_k$. We highlight that the CEP graphical model in Fig. 10 and in Fig. 3 are equivalent. The model in Fig. 10, represent the CEP as the distribution maximizing a set of rewards $E_0, \dots\ E_k$, while the model in Fig. 3 represent the CEP as the distribution maximizing the combination of a set of policies. Depending on the problem, we might be interested to frame the problem with one or the other approach.

APPENDIX C
COMPOSABLE ENERGY POLICIES AND ARTIFICIAL POTENTIAL FIELDS METHOD

In the following section, we will show that the Artificial Potential Fields Method can be written as a particular case of Composable Energy Policies.

Artificial Potential Field Method assumes a weighted sum of the accelerations generated by a set of dynamic components

$$\ddot{\boldsymbol{q}} = \sum_{k=0}^{K} \boldsymbol{\Lambda}_k \boldsymbol{g}_k(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{57}$$

Let's consider a particular case of CEP were all the components are represented with a normal distribution centered in $\boldsymbol{g}_k(\boldsymbol{q}, \dot{\boldsymbol{q}})$,

$$\pi_k(\ddot{\boldsymbol{q}}|\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{N}(\boldsymbol{g}_k(\boldsymbol{q}, \dot{\boldsymbol{q}}), \Sigma_k). \tag{58}$$

Then, given K components

$$\pi(\ddot{\boldsymbol{q}}|\boldsymbol{q}, \dot{\boldsymbol{q}}) = \prod_{k=0}^{K} \mathcal{N}(\boldsymbol{g}_k(\boldsymbol{q}, \dot{\boldsymbol{q}}), \Sigma_k) = \mathcal{N}(\boldsymbol{g}_\Sigma(\boldsymbol{q}, \dot{\boldsymbol{q}}), \Sigma_\Sigma) \tag{59}$$

with

$$\boldsymbol{g}_\Sigma = \left(\sum_{k=0}^{K} \Sigma_k^{-1}\right)^{-1} \left(\sum_{k=0}^{K} \Sigma_k^{-1} \boldsymbol{g}_k\right) \tag{60}$$

and

$$\Sigma_\Sigma = \left(\sum_{k=0}^{K} \Sigma_k^{-1}\right)^{-1}. \tag{61}$$

In CEP, the acceleration is obtained by an optimization of

$$\ddot{\boldsymbol{q}} = \arg\max_{\ddot{\boldsymbol{q}}} \pi(\ddot{\boldsymbol{q}}|\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{62}$$

The maximum of 59 is known; given the policy is a gaussian, the maximun is the mean in 60

$$\ddot{\boldsymbol{q}} = \left(\sum_{k=0}^{K} \Sigma_k^{-1}\right)^{-1} \left(\sum_{k=0}^{K} \Sigma_k^{-1} \boldsymbol{g}_k(\boldsymbol{q}, \dot{\boldsymbol{q}})\right). \tag{63}$$

We can rewrite (63) to (57)

$$\boldsymbol{\Lambda}_k = \left(\sum_{k=0}^{K} \Sigma_k^{-1}\right)^{-1} . \Sigma_k^{-1} \tag{64}$$

We show that APF policy can be rewritten as a CEP, modelling the dynamics by a product of normal distributions. Modelling all the components by normal distributions might be a bad choice for a proper modules integration. normal distributions assume that (1) there is a unique optimal action (the mean) to solve the task and (2) the actions quality 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.

APPENDIX D
EXPERIMENTS

A. Reactive Motion Generation

The modular components (Go-to, Obstacle Avoidance and joint limits avoidance) for both baselines APF and RMP were modeled based on [20] and [3] respectively. CEP were modeled following the energies presented in section IV. We introduce a table with the hyperparameters for all CEP energies

B. Prior based policies for RL

We introduce a Table with the Hyperparameters for the PPO in Mushroom-RL [6]. We used the same hyperparameters for the three methods (Behavioural Cloning + RL, Residual RL and CEP for RL).

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 III
COMPONENT PARAMETERS FOR COMPOSABLE ENERGY POLICIES IN REACTIVE MOTION GENERATION

Hyperparameters	
horizon	1400
Policy Net	18-128-128-3
policy std0	1.
policy batch	256
n_epochs_policy	2
policy learn rate	1e-4
Critic Net	18-256-256-1
critic batch	256
critic learn rate	3e-4
n_steps_per_fit	horizon x 30
Discount factor	.997

TABLE IV
COMPONENT PARAMETERS FOR COMPOSABLE ENERGY POLICIES IN REACTIVE MOTION GENERATION