Hierarchical Policy Blending As Optimal Transport

An T. Le AN.LE@TU-DARMSTADT.DE
Computer Science Department, TU Darmstadt, Germany

Kay Hansel KAY.HANSEL@TU-DARMSTADT.DE
Computer Science Department, TU Darmstadt, Germany

Jan Peters JAN.PETERS@TU-DARMSTADT.DE
Computer Science Department, TU Darmstadt, Germany
DFKI, RD SAIRoL, Hessian.AI, Centre for Cognitive Science

Georgia Chalvatzaki GEORGIA.CHALVATZAKI@TU-DARMSTADT.DE
Computer Science Department, TU Darmstadt, Germany and Hessian.AI

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

We present hierarchical policy blending as optimal transport (HiPBOT). HiPBOT hierarchically adjusts the weights of low-level reactive expert policies of different agents by adding a look-ahead planning layer on the parameter space. The high-level planner renders policy blending as unbalanced optimal transport consolidating the scaling of the underlying Riemannian motion policies. As a result, HiPBOT effectively decides the priorities between expert policies and agents, ensuring the task's success and guaranteeing safety. Experimental results in several application scenarios, from low-dimensional navigation to high-dimensional whole-body control, show the efficacy and efficiency of HiPBOT. Our method outperforms state-of-the-art baselines – either adopting probabilistic inference or defining a tree structure of experts – paving the way for new applications of optimal transport to robot control. More material at https://sites.google.com/view/hipobot.

Keywords: Reactive Motion Generation, Optimal Transport, Riemannian Motion Policies

1. Introduction

Reactive motion generation is a fundamental functionality for robots operating in complex and unstructured settings, where dynamic changes can occur (Park et al., 2008; Hogan and Sternad,

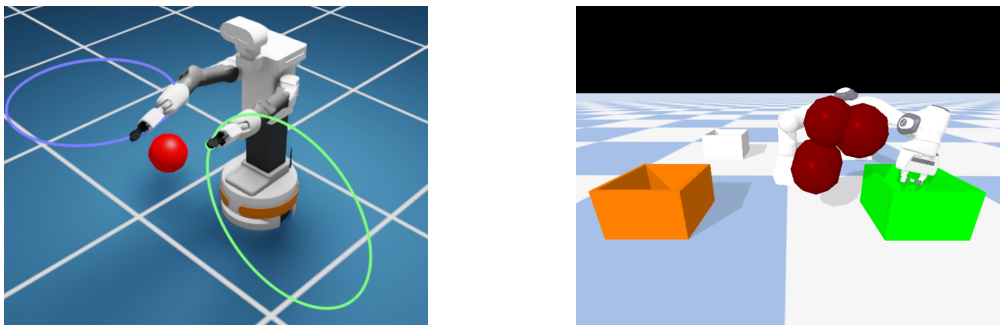

Figure 1: (Left) TIAGo++ whole-body control environment. The two arms of TIAGo++ are treated as two agents tracking different reference trajectories while avoiding collisions. (Right) Panda manipulation environment. The Panda has to start from a randomly selected box (brown) and reach the randomly selected target box (green) while avoiding all possible collisions with itself and the dynamic environment (red obstacles can be static or moving with constant velocity).

2012; Ijspeert et al., 2013; Paraschos et al., 2018; Ratliff et al., 2018). Reactivity emerges through high-frequency control policies that can resolve a specific behavior, e.g., approaching a goal, or avoiding an obstacle (Khatib, 1987) or combinations of such behaviors (Cheng et al., 2018).

Typical methods in literature achieve such reactivity through three key approaches. First, operational space control (OSC) with a hierarchy of tasks (Khatib, 1987; Khatib et al., 2004), where a sequence of quadratic programming (QP) problems is resolved in the nullspace of the previous task in the priority list (Flacco et al., 2015). Second, reactive motion generation through the synthesis of Riemannian Motion Policies (RMPs) defined in a tree structure (Cheng et al., 2018, 2021), which generalizes the OSC framework to geometric dynamical systems (GDS) (Bullo and Lewis, 2019). The third approach refers to learning-based approaches that use either demonstrated data – imitation learning (Ijspeert et al., 2013; Paraschos et al., 2018)– or trial-and-error interactions with the environment – reinforcement learning (RL) (Kober et al., 2013; Moos et al., 2022) – for learning to react and adapt to domain changes.

On the opposite side of the spectrum lie planning-based approaches – either sampling-based (Kavraki et al., 1996; LaValle, 1998, 2006; Kalakrishnan et al., 2011; Bhardwaj et al., 2022) or gradient-based (Zucker et al., 2013; Mukadam et al., 2018) – which simulate a look-ahead forward modeling of the environment, towards reaching a predetermined goal. The interplay of planning and reactivity is long studied in robotics (Kaelbling and Lozano-Pérez, 2010; Srivastava et al., 2014; Pertsch et al., 2020; Bahl et al., 2021), and it frequently emerges when a high-level agent sets subgoals towards a long-term goal for a myopic reactive agent to achieve. This hierarchical planning and control can be realized by explicitly setting task sub-goals (Jauhri et al., 2022; Xia et al., 2020; Sharma et al., 2021), or by adapting parameters of the low-level policy towards a single long-term goal (Kroemer et al., 2015; End et al., 2017; Celik et al., 2022; Akrour et al., 2021; Zaki et al., 2022).

Our work focuses on the latter aspect of hierarchical control. We present a planning approach on the parameter space of given reactive policies describing different robot behaviors. Representing the low-level policies in exponential form allows a seamless composition in the form of a product of experts. We frame the look-ahead planning as an Unbalanced Optimal Transport (UOT) problem redistributing the multimodal weights of the expert policies. This formulation enables the simultaneous adaptation of expert policies utilized by multiple agents. Hence, our approach consolidates the low-level experts of various agents and helps in practice to avoid local minima in highly dynamic environments.

To summarize, our contributions are **(i)** introducing the definition of multi-expert-multi-agent blending problem, **(ii)** casting the blending problem as entropic-regularized UOT, thereby utilizing its convexity and efficient solver. And, **(iii)** we show that by realizing experts as RMPs, we can maintain asymptotic local stability.

2. Preliminaries And Problem Statement

We introduce preliminaries for defining stable experts for reactive motion generation. Then, we state the policy-blending problem by representing the experts in exponential form and cast it as an entropic-regularized UOT.

2.1. Riemannian Motion Policy

An RMP (Ratliff et al., 2018) is a mathematical object $(\boldsymbol{\pi}, \boldsymbol{M})$ representing reactive, modular, and composable motion generation policies, where $\boldsymbol{\pi}$ is a deterministic policy mapping states to ac-

tions, and $\boldsymbol{M}$ is the Riemannian matrix representing the policy weight. The state $\boldsymbol{s} = (\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{c})$ represents the robot's position $\boldsymbol{q} \in \mathbb{R}^q$, velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^q$ and environment context $\boldsymbol{c}$. We assume a set of homeomorphic task maps $\{\phi_i : \mathcal{Q} \to \mathcal{X}_i\}$, that relate the robot configuration $\mathcal{Q}$ space and a certain task space $\mathcal{X}_i$ of the i^{th} task. Then, given a set of task-space policies $(\boldsymbol{\pi}^{\mathcal{X}_i}, \boldsymbol{M}^{\mathcal{X}_i})$, we can represent a deterministic acceleration policy in the robot configuration space by $\boldsymbol{\pi} = \boldsymbol{a} = \ddot{\boldsymbol{q}} = \boldsymbol{M}^{\dagger} \sum_i \boldsymbol{J}_{\phi_i}^{\intercal} \boldsymbol{M}^{\mathcal{X}_i} \boldsymbol{\pi}^{\mathcal{X}_i}(\phi(\boldsymbol{s}))$, with $\boldsymbol{M} = \sum_i \boldsymbol{J}_{\phi_i}^{\intercal} \boldsymbol{M}^{\mathcal{X}_i} \boldsymbol{J}_{\phi_i}$, $\boldsymbol{J}_{\phi_i}$ the Jacobian of the task map ϕ_i and $\dagger$ is the pseudo-inverse operator.

2.2. Product of Experts

To define the policy blending problem, we formalize each expert policy $i \in \{1, \dots, n\}$ in a set of n policies as the Boltzmann distribution form $\pi_i(\boldsymbol{a} \mid \boldsymbol{s}; \boldsymbol{\theta}_i) \propto \exp(-E_i(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_i))$, where the quantities $\boldsymbol{s} \in \mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}$ denote a state and an action, respectively. An energy function $E_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a cost to each state-action pair. The choice of the energy function E_i and its hyperparameter $\boldsymbol{\theta}_i$ is usually designed or learned in advance. Following the PoE (Hinton, 2002) formulation, the blended policy for an agent can be defined as

$$\pi(\boldsymbol{a} \mid \boldsymbol{s},\ \boldsymbol{\beta}) = \prod_{i=1}^{n} \pi_i(\boldsymbol{a} \mid \boldsymbol{s};\ \boldsymbol{\theta}_i)^{\beta_i} \propto \exp\left(-\sum_{i=1}^{n} \beta_i E_i(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_i)\right) \tag{1}$$

with blending/weighting factors $\boldsymbol{\beta}$, also known as *temperatures*, representing the importance or relevance of each policy in the product. In the logarithmic space, this policy blending corresponds to a weighted superposition by performing Maximum Likelihood (MLE) on the PoE $\boldsymbol{a}^* = \operatorname{argmax}_{\boldsymbol{a} \in \mathcal{A}} \log \pi(\boldsymbol{a} \mid \boldsymbol{s}, \boldsymbol{\beta}) = \operatorname{argmin}_{\boldsymbol{a} \in \mathcal{A}} \sum_{i=1}^{n} \beta_i E_i(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_i)$ depending on state $\boldsymbol{s}$ and $\boldsymbol{\beta}$. We can further formulate the state-dependent temperature $\boldsymbol{\beta}(\boldsymbol{s})$, giving the possibility to change the state-dependent relevance or importance of experts. In an online fashion, a change in the expert weighting (i.e., expert relevancy) makes it possible to induce *parameter planning* into the myopic nature of the policy $\pi(\boldsymbol{a} \mid \boldsymbol{s})$. In particular, the simplest form of such parameter planning problem (i.e., a policy blending problem in our paper) can be formulated as the linear program $\min_{\boldsymbol{\beta} \in \mathbb{R}_+^n} \langle \boldsymbol{\beta}, \boldsymbol{C}(\boldsymbol{s}) \rangle$, with $\langle \cdot, \cdot \rangle$ is the Frobenius dot-product, and a state-dependent objective matrix $\boldsymbol{C}(\boldsymbol{s})$ (e.g., rollout return in RL settings) dictating the situational blending weights $\boldsymbol{\beta}$. We realize the PoE in the RMP framework utilizing its stability property, as shown in next sections.

2.3. Preliminaries On Optimal Transport

We give a brief introduction to OT and motivate why solving the linear program of policy blending using OT.

Histograms And Transport Polytope. For two histograms $\boldsymbol{n} \in \Sigma_n$ and $\boldsymbol{m} \in \Sigma_m$ in the simplex $\Sigma_d \overset{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}_+^d : \boldsymbol{x}^{\intercal} \mathbf{1}_d = 1\}$, we define the transport polytope $U(\boldsymbol{n}, \boldsymbol{m})$, namely the set of $n \times m$ matrices $U(\boldsymbol{n}, \boldsymbol{m}) \overset{\text{def}}{=} \{\boldsymbol{P} \in \mathbb{R}_+^{n \times m} \mid \boldsymbol{P}\mathbf{1}_m = \boldsymbol{n}, \boldsymbol{P}^{\intercal}\mathbf{1}_n = \boldsymbol{m}\}$ where $\mathbf{1}_d$ is the d-dimensional vector of ones. From a probabilistic view, the set of $U(\boldsymbol{n}, \boldsymbol{m})$ contains all possible *joint probabilities* of two multinomial random variables (X, Y) having histograms $\boldsymbol{n}$ and $\boldsymbol{m}$, respectively. Indeed, any matrix $\boldsymbol{P} \in U(\boldsymbol{n}, \boldsymbol{m})$ can be identified as the joint probability for (X, Y) such that $p(X = i, Y = j) = p_{ij}$. We define the entropy $H(\cdot)$ of these histograms and their marginals as $\boldsymbol{x} \in \Sigma_n$, $H(\boldsymbol{x}) = -\sum_{i=1}^{n} x_i \log x_i$, $H(\boldsymbol{P}) = -\sum_{i,j=1}^{n,m} p_{ij}(\log p_{ij} - 1)$.

Entropic-Regularized Optimal Transport. Given a $n \times m$ cost matrix $\boldsymbol{C}$, the OT between $\boldsymbol{n}$ and $\boldsymbol{m}$ given cost $\boldsymbol{C}$ is $d_C(\boldsymbol{n}, \boldsymbol{m}) \overset{\text{def}}{=} \min_{\boldsymbol{P} \in U(\boldsymbol{n}, \boldsymbol{m})} \langle \boldsymbol{P}, \boldsymbol{C} \rangle$, which is exactly the above policy blending

problem with additional histogram constraints. However, solving this linear program is expensive for large matrix dimensions in our reactive motion generation setting. For a general matrix $\boldsymbol{C}$ and $d = \max(n, m)$, the worst case complexity of classical algorithms (Ahuja et al., 1988; Orlin, 1988) solving for the optimal plan $\boldsymbol{P}^*$ is $O(d^3 \log d)$. To deal with the scalability of computing the OT, Cuturi (2013) propose to regularize its objective function by the entropy of the transport plan, which results in the entropic-regularized OT with an entropy scaling scalar λ

$$\boldsymbol{P}^{\lambda} \stackrel{\text{def}}{=} \underset{\boldsymbol{P} \in U(\boldsymbol{n}, \boldsymbol{m})}{\operatorname{argmin}} \langle \boldsymbol{P}, \boldsymbol{C} \rangle - \lambda H(\boldsymbol{P}). \tag{2}$$

The solution $\boldsymbol{P}^{\lambda}$ is unique due to the strict convexity of the negative entropy term. The entropic regularization enables the celebrated Sinkhorn-Knopp algorithm (Sinkhorn, 1967) to solve OT, shown to have a complexity of $\tilde{O}(d^2/\epsilon^3)$ (Altschuler et al., 2017), with ϵ is the approximation error.

Entropic-Regularized Unbalanced Optimal Transport. A key constraint of classical OT is that it requires the input measures to be normalized to the unit mass, which is a problematic assumption for many applications that require handling arbitrary positive measures (mass creation or destruction), and/or allowing for only partial displacement of mass. The entropic-regularized UOT (Frogner et al., 2015; Chizat et al., 2018) is defined as

$$\boldsymbol{P}^{\lambda}_{UOT} \stackrel{\text{def}}{=} \underset{\boldsymbol{P} \in \mathbb{R}^{n \times m}_{+}}{\operatorname{argmin}} \langle \boldsymbol{P}, \boldsymbol{C} \rangle - \lambda H(\boldsymbol{P}) + \lambda_{KL} \left(\widetilde{\mathrm{KL}}(\boldsymbol{P}\mathbf{1}_m \parallel \boldsymbol{n}) + \widetilde{\mathrm{KL}}(\boldsymbol{P}^{\intercal}\mathbf{1}_n \parallel \boldsymbol{m}) \right) \tag{3}$$

where now $\boldsymbol{n} \in \mathbb{R}^n_+, \boldsymbol{m} \in \mathbb{R}^m_+$ are arbitrary positive vectors, λ_{KL} is the marginal regularization scalar, and $\widetilde{\mathrm{KL}}(\boldsymbol{w}||\boldsymbol{z}) = \boldsymbol{w}^{\intercal} \log(\boldsymbol{w} \oslash \boldsymbol{z}) - \mathbf{1}^{\intercal}\boldsymbol{w} + \mathbf{1}^{\intercal}\boldsymbol{z}$ is the generalized Kullback-Leibler (KL) divergence between two positive vectors $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^k_+$ ($\oslash$ is the element-wise division), with the convention $0 \log 0 = 0$. Pham et al. (2020) show that the (Sinkhorn-like) generalized matrix scaling algorithm (Frogner et al., 2015) solves the dual of (3) with the complexity of $\tilde{O}(d^2/\epsilon)$ and is guaranteed to converge (Theorem 4.1 in (Chizat et al., 2018)). With these properties, casting policy blending as an entropic-regularized UOT problem is desirable since UOT relaxes the normalizing constraint, bringing the OT problem back to the linear program form of policy blending, as policy blending weights are usually unnormalized quantities. Moreover, assuming a Gaussian distribution over blending weights is often non-realistic, as multiple policies may have similar priorities given the current situation. Finally, the entropic-regularized UOT benefits from the computation efficiency of the Sinkhorn-like algorithm.

3. Hierarchical Policy Blending As Optimal Transport

In this section, we propose **Hi**erarchical **P**olicy **B**lending as **O**ptimal **T**ransport (HiBPOT) - a two-level hierarchical scheme for reactive motion generation. We hypothesize there exist multiple agents controlling the same dynamical system satisfying some objectives at the upper-level, where each agent utilizes myopic (learned or crafted) expert policies to compute their actions at the lower-level (Hansel et al., 2022). In particular, each agent can choose to control a subset or all DoFs of the dynamical system. The upper level employs entropic-regularized UOT to solve the policy blending problem by observing expert rollouts that inform the weight-scaling of the lower-level agents.

3.1. Product Of Experts-Agents

Let us consider multi-arm systems (Fig. 1-left), where each robotic arm can be considered an agent acting on the whole system to execute some tasks. Assuming the agents' behaviors are collaborative and that there exists a pool of n experts and m agents, we propose a simple solution for the Multi-Experts-Multi-Agents (MEMA) policy blending problem by extending the PoE (1), as defined in Definition 1.

Definition 1 (Product of Experts-Agents) *Let $\boldsymbol{\beta} \in \mathbb{R}_+^{n\times m}$ be the positive blending weight matrix for n experts and m agents. The MEMA blending policy is defined as the product of experts-agents (PoEA)*

$$\pi(\boldsymbol{a} \mid \boldsymbol{s},\ \boldsymbol{\beta}) = \prod_{i,j=1}^{n,m} \pi(\boldsymbol{a}_{i,j} \mid \boldsymbol{s};\ \boldsymbol{\theta}_{i,j})^{\beta_{i,j}} \propto \exp\left(-\sum_{i,j=1}^{n,m} \beta_{i,j} E(\boldsymbol{s}, \boldsymbol{a}_{i,j};\ \boldsymbol{\theta}_{i,j})\right) \tag{4}$$

with i, j index the i^{th}-expert and j^{th}-agent, $\boldsymbol{s}$ is the holistic system state observed by all experts, and $\boldsymbol{a}$ is the blended pullbacked action from all experts and agents.

We realize the lower-level experts within the RMP framework. In RMP, for the i^{th}-expert of j^{th}-agent, the task-space energy $E(\boldsymbol{x}, \boldsymbol{a}_{i,j}; \boldsymbol{\theta}_{i,j})$ is usually designed as a quadratic function having smooth and convex properties in the task space $\boldsymbol{x} \in \mathcal{X}_{i,j}$, $\boldsymbol{a}_{i,j} \in \mathcal{A}_{i,j}$, with corresponding task map $\boldsymbol{x} = \phi_{i,j}(\boldsymbol{s})$. Accordingly, the Boltzmann distribution forms a Gaussian $\pi(\boldsymbol{a}_{i,j} \mid \boldsymbol{x}; \boldsymbol{\theta}_{i,j}) = \mathcal{N}(\boldsymbol{M}_{i,j}(\boldsymbol{x})^{-1}\boldsymbol{f}_{i,j}(\boldsymbol{x}), \boldsymbol{M}_{i,j}(\boldsymbol{x})^{-1})$ locally at $\boldsymbol{x}$ with the forcing function $\boldsymbol{f}_{i,j}(\boldsymbol{x})$ and $\boldsymbol{M}_{i,j}(\boldsymbol{x})$ as the mean and the Riemannian matrix (i.e., the precision matrix), respectively. Within the PoEA view, the *pullbacked* forcing term and Riemannian matrix of j^{th}-agent's configuration policy would be $\boldsymbol{f}_j(\boldsymbol{s}) = \sum_{i=1}^n \beta_{i,j}(\boldsymbol{s}) \boldsymbol{J}_{\phi_{i,j}}^\intercal \boldsymbol{f}_{i,j}(\boldsymbol{x})$, $\boldsymbol{M}_j = \sum_{i=1}^n \beta_{i,j}(\boldsymbol{s}) \boldsymbol{J}_{\phi_{i,j}}^\intercal \boldsymbol{M}_{i,j}(\boldsymbol{x}) \boldsymbol{J}_{\phi_{i,j}}$, respectively. Given the current state and determined temperatures, the MLE blended action (at configuration space) can be computed analytically in closed form (Ratliff et al., 2018) as

$$\boldsymbol{a}^* = \operatorname*{argmin}_{\boldsymbol{a}\in\mathcal{A}} \sum_{i,j=1}^{n,m} \beta_{i,j} E(\boldsymbol{s}, \boldsymbol{a}_{i,j};\ \boldsymbol{\theta}_{i,j}) = \sum_j \boldsymbol{M}_j^\dagger \boldsymbol{f}_j(\boldsymbol{s}). \tag{5}$$

3.2. Policy Blending As Entropic-Regularized Unbalanced Optimal Transport

In practice, we found that the normalizing constraint of the policy temperature is restrictive, as it requires spreading enough masses to the policy weight-scaling, leading to overestimation. On the other hand, in case of large number of experts, the normalized temperature matrix puts too small masses on expert policies, thus leading to underestimation and making them underperforming. Thus, we propose to cast policy blending as an entropic-regularized UOT problem to relax the need for the normalizing constraint.

Definition 2 (MEMA Policy Blending) *Let $\boldsymbol{n} \in \mathbb{R}_+^n, \boldsymbol{m} \in \mathbb{R}_+^m$ be arbitrary positive vectors representing the priors of expert-policy and agent-policy temperatures, respectively. The entropic-regularized UOT for the policy blending is defined as*

$$\boldsymbol{\beta}^*(\boldsymbol{s}) = \operatorname*{argmin}_{\boldsymbol{\beta}\in\mathbb{R}_+^{n\times m}} \langle \boldsymbol{\beta}, \boldsymbol{C} \rangle - \lambda H(\boldsymbol{\beta}) + \lambda_{KL}\left(\widetilde{\mathrm{KL}}(\boldsymbol{\beta}\mathbf{1}_m \parallel \boldsymbol{n}) + \widetilde{\mathrm{KL}}(\boldsymbol{\beta}^\intercal \mathbf{1}_n \parallel \boldsymbol{m})\right) \tag{6}$$

with $\boldsymbol{C}(\boldsymbol{s})$ is the state-dependent cost matrix, which can be learned or computed analytically.

Algorithm 1: HiPBOT

Input: Goal set $\{\boldsymbol{s}_j^g\}$, transition function $\mathcal{T}(\cdot,\cdot)$, signed-distance field set $\{\text{SDF}_j(\cdot)\}$, UOT parameters $\{\lambda, \lambda_{KL}\}$, cost weights $\{w_g, w_c\}$, expert parameter set $\{\boldsymbol{\theta}_{i,j}\}$

while *Not All Goals Reached* **do**

  Update $\boldsymbol{s}_j^g$ and $\{\text{SDF}_j(\cdot)\}$ given some perception model.

  for $i = 1, 2, ..., n$ *and* $j = 1, 2, ..., m$ **do in batch**

    Do rollout and compute $[\boldsymbol{C}(\boldsymbol{s})]_{i,j}$ as in Eq. (7).

  end

  Solve $\boldsymbol{\beta}(\boldsymbol{s})$ (Definition 1) with Sinkhorn-like algorithm Chizat et al. (2018).

  `// Compute agent optimal action`

  $\boldsymbol{a}^* = \text{argmin}_{\boldsymbol{a}\in\mathcal{A}} \sum_{i,j=1}^{n,m} \beta_{i,j}^*(\boldsymbol{s}) E(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta}_{i,j})$

  `// Rollout new state`

  $\boldsymbol{s} = \mathcal{T}(\boldsymbol{s}, \boldsymbol{a}^*)$

end

The solution of (6) is unique due to the strict convexity of the objective in $\boldsymbol{\beta}$. Due to the uniqueness of the solution and the practical computation complexity of the Sinkhorn-like algorithm solving (6), it is well-suited for reactive motion generation. Note that this formulation optimizes the blending temperatures depending on the objective costs at the upper-level, while still assuming expert independency at the lower-level.

In the dynamics settings of motion planning, the objectives are usually goal-reaching, obstacle avoidance, and self-collision avoidance in dynamic settings. Hence, we follow these objectives to design the state-dependent cost matrix as

$$[\boldsymbol{C}(\boldsymbol{s})]_{i,j} = \frac{1}{h}\sum_{t=1}^{h} w_g \underbrace{d(\boldsymbol{s}_{i,j}^t, \boldsymbol{s}_j^g)}_{\text{Goal Cost}} + w_c \underbrace{\exp\left(-\frac{\text{SDF}_j(\boldsymbol{s}_{i,j}^t)^2}{2l^2}\right)}_{\text{Collision-Avoidance Cost}} \tag{7}$$

where from the current system state $\boldsymbol{s}$, the rollout with horizon h from the perspective of j^{th}-agent following the i^{th}-expert is $\{\boldsymbol{s}, \boldsymbol{s}_{i,j}^1, \ldots, \boldsymbol{s}_{i,j}^h\}$. We assume the transition dynamics $\mathcal{T}(\cdot,\cdot)$ of the system known, and an expert rollout is computed by following $\boldsymbol{s}^{t+1} = \mathcal{T}(\boldsymbol{s}^t, \boldsymbol{a}^*)$, $\boldsymbol{a}^* = \text{argmin}\, E(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})$. $\boldsymbol{s}_j^g$ is the j^{th}-agent goal state, and $\text{SDF}_j(\cdot)$ is the signed distance field measuring the closest distance of the j^{th}-agent's robot links to obstacles including itself (i.e., self-distance). l is the hyperparameter for the collision margin. Note that the goal or obstacles can be changed over time; thus, the goal and distance field are also updated in the loop. This cost design enables integration of additional higher levels of planning abstractions, e.g., task planning, where the symbolic planner can set the intermediate goals or other contexts in the cost matrix (but this is not integrated in the current work). Since experts are independent by assumption at the lower level, the elements of the cost matrix can be computed in batch using GPU (Algorithm 1).

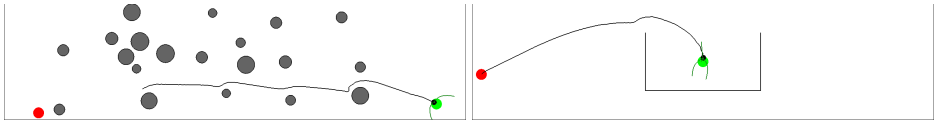

Figure 2: (Left) Planar Maze Environment. We randomly sample K circular obstacles inside a restricted area between the start and goal positions. We model the movement of the obstacles using simple Euler integration. (Right) Planar Box Environment. The box can be either static or dynamic, and its motion is modeled as a constant velocity. In both environments, an agent moves from a random start (red) to a random goal (green) position. Green lines are expert rollouts.

3.3. Stability Analysis

HiPBOT, as we deploy as expert policies RMPs, only sets the scaling factor for both $\boldsymbol{f}_{i,j}, \boldsymbol{M}_{i,j}$ at the lower-level. Analyzing its local stability is straightforward.

Proposition 3 (Asymptotic Stability) *As $\boldsymbol{\beta} \in \mathbb{R}_{+}^{n \times m}$ by Definition 2 is positive, if all expert RMPs are in the form of Geometric Dynamical Systems, then by **Theorem 2** in (Cheng et al., 2018), the system that follows the HiPBOT policy as Product of Experts-Agents in Definition 1 converges to the forward invariance set $\mathcal{C}_{\infty} := \{(\boldsymbol{s}, \dot{\boldsymbol{s}}) : \boldsymbol{f}(\boldsymbol{s}) = 0, \dot{\boldsymbol{s}} = 0\}$.*

Note that this local stability of HiPBOT is only valid for static environments, where the parameters for collision-avoidance RMPs do not change. Nevertheless, for dynamic environments, in most cases, we empirically observed that the agents also exhibit locally stable behaviors, and we plan to investigate theoretically further in the future.

4. Experiments

We evaluate HiPBOT in two toy environments and a manipulation task with a 7DoF robot. These settings are multi-experts-one-agent. Finally, we demonstrate HiPBOT in whole-body reactive motion generation.

Baselines. We benchmark HiPBOT against two baselines. First, RMPflow (Cheng et al., 2018), a myopic baseline without look-ahead evaluation, composes all expert RMPs to generate a global dynamical behavior. It also runs at a very high frequency due to low computational demands. Second, as a strong baseline, HiPBI (Hansel et al., 2022) is a similar hierarchical scheme to our HiPBOT, which adopts probabilistic inference to address the blending problem. HiPBI samples the rollouts from a temperature proposal distribution and updates the temperature distribution in an online fashion, while HiPBOT shoots individual experts evaluating their contributions assuming unnormalized temperature priors. For both HiPBOT and HiPBI, the environment and the methods operate asynchronously. The algorithms must make quick decisions to react to unexpected changes in the landscape.

Metrics and Settings. We use the following metrics: (i) success rate (SUC), indicating the percentage of goal reaching without any collisions; (ii) safety rate (SAFE) of collision-free motions regardless of reaching the goal or not; (iii) the final l2 distance (l2D) to the goal; and (iv) the total time steps (TS) need until the goal is reached. For a comparative study, we use different rollout horizons for HiPBI and HiPBOT.

4.1. Toy Environments

The **Planar Maze Environment** is a cluttered environment, see Fig. 2. This maze environment mimics a dense, cluttered, and dynamic environment. In this case, local minima are created but

often disappear independently. However, the control methods have to be reactive enough to avoid collisions. Unlike the maze, the **Planar Box Environment** is a sparse domain. The agent start position is sampled randomly to the right or left of the box. The challenge lies in not getting into a local optimum in front (left or right) or below the box. Furthermore, the dynamic nature complicates the planning of a promising solution. In this case, although the reactive requirement is relaxed due to being sparse, the difficult local minima always exist. We design common RMPs such as collision avoidance and goal reaching for all methods, as in (Cheng et al., 2018). To achieve a curving behavior, we design an expert $\pi_{\mathbf{curl}}$ that exerts forces on the normal space of the potential forces and add two opposing curling experts for balancing. Although these curving experts do not affect RMPflow, hierarchical methods achieve curving behavior by adapting blending weights.

Comparative Evaluation. Table 1 shows the comparative results for the static versions of the toy environments. It is evident that the myopic RMPflow is not able to solve the Box domain, but it guarantees safety due to its stable property. Short horizons in HiPBI and HiPBOT are not as effective as longer ones. It is notable that HiPBOT, with only 10 steps look-ahead, outperforms the baselines in most metrics, guaranteeing maximum safety and good success rates. Table 2, shows the dynamic versions of the toy tasks for both synchronous (S) and asynchronous (A) execution of policy blending. HiPBOT outperforms the baselines in all cases with short horizons, making it much faster for deployment in highly dynamic domains. With comparative performance, considering the rollout computation and optimization, we observe that HiPBOT ($h = 10$) achieves mean planning rate of around 30Hz due to being efficient with shorter horizon, while HiPBI ($h = 50$) runs at about 2Hz. This computing gap hurts the performance of baselines even more in highly dynamic environments, as seen in Fig. 3, depicting a *stress-test* on the 2D Maze for changing acceleration levels of the obstacles by adding Brownian noises. We evaluate a plain goal-reaching success rate–regardless of collisions, along with the safety rate and collision-free success rate. Evidently, HiPBOT with $h = 10$ performs overall better, even in the extreme scenarios [1].

Stress-test of HiPBOT. We even tested more difficult scenarios for the HiPBOT ($h = 10$) in the 2D Maze environment, where we varied both numbers of dynamic obstacles and their acceleration levels. While there is total success regardless of collisions, safety is compromised in these extreme dynamic cases. We hypothesize that longer horizons, more efficient optimization, and even a "cleverer" exploration strategy during planning are necessary for these more complex environments.

4.2. Manipulator Environment

We test the scalability of our method in high-dimensional manipulation tasks, Fig. 1. We implemented eight expert RMPs, ranging from joint & velocity limits, self-collision to reaching the target, and obstacle avoidance. Both HiPBI and HiPBOT use the same experts with four additional local curling experts for the end-effector that operates in the normal space of the target-reaching potential. While RMPflow cancels out the curling, hierarchical methods adapt the temperatures to achieve the desired dynamic behavior. HiPBOT with horizon 10 is again superior in terms of collision-free success rate. We see that performance drops in the static environment due to difficult local minima that do not vanish over time. This case would require longer look-ahead horizons, but we want to point to the increased efficiency with as few as 10 steps with mean planning rate of 6.4Hz, compared to the 50 steps of HiPBI with mean planning time of 3.5s, which yields lower performance.

1. See videos in our project website https://sites.google.com/view/hipobot/experiment-videos

Table 1: Evaluation of HiPBOT versus baselines on the static planar environments. This experiment shows the capabilities of HiPBOT to overcome local minima (100 seeds per evaluation).

	2D Toy Box Environment				2D Toy Maze Environment			
	SUC[%]	SAFE[%]	D2G	TS	SUC[%]	SAFE[%]	D2G	TS
RMP*flow*	0	**100**	198.8 ± 0.7	500.0 ± 0.0	73	99	161.5 ± 296.7	338.7 ± 100.8
HiPBI ($h = 25$)	0	**100**	198.9 ± 0.5	500.0 ± 0.0	77	93	148.2 ± 284.5	331.3 ± 98.9
HiPBI ($h = 5$)	64	**100**	82.1 ± 79.9	354.3 ± 169.7	77	97	151.1 ± 284.1	**323.2 ± 99.3**
HiPBOT ($h = 5$)	0	**100**	176.2 ± 1.2	500.0 ± 0.0	72	96	200.0 ± 334.2	386.2 ± 234.2
HiPBOT ($h = 10$)	**93**	**100**	**17.2 ± 6.7**	**132.9 ± 10.1**	**82**	**100**	**138.5 ± 300.6**	401.0 ± 281.6

Table 2: Evaluation of HiPBOT vs. baselines on the dynamic planar environments with 10-pixel velocity levels. We also compare HiPBI and HiPBOT in synchronous (S) and asynchronous (A) settings. In (S), the simulation waits for the blending solution before the agent steps in the environment. In (A), the environment and the methods act asynchronously. This experiment demonstrates the computational advantage of HiPBOT with short look-ahead horizon, balancing between being reactive and explorative for safety (100 seeds per evaluation).

	2D Toy Box Environment				2D Toy Maze Environment			
	SUC[%]	SAFE[%]	D2G	TS	SUC[%]	SAFE[%]	D2G	TS
RMP*flow*	0	**100**	198.9 ± 1.5	500.0 ± 0.0	77	89	161.5 ± 620.0	330.7 ± 191.3
HiPBI ($h = 25$, S)	2	**100**	189.3 ± 44.7	490.9 ± 81.8	98	**99**	20.3 ± 172.7	247.6 ± 55.8
HiPBI ($h = 50$, S)	61	**100**	49.5 ± 75.6	276.6 ± 251.2	**99**	**99**	17.5 ± 162.6	**247.5 ± 47.6**
HiPBI ($h = 75$, S)	**100**	**100**	**7.3 ± 5.9**	131.9 ± 18.0	**99**	**99**	**19.0 ± 171.7**	252.1 ± 47.3
HiPBOT ($h = 5$, S)	0	**100**	199.3 ± 1.1	500.0 ± 0.0	**99**	**99**	26.1 ± 108.8	315.9 ± 129.5
HiPBOT ($h = 10$, S)	**100**	**100**	25.8 ± 0.6	143.9 ± 22.8	**99**	**99**	22.0 ± 72.9	294.2 ± 108.1
HiPBOT ($h = 15$, S)	**100**	**100**	16.4 ± 4.1	**127.3 ± 18.2**	98	98	30.8 ± 104.5	312.9 ± 145.6
HiPBI ($h = 25$, A)	7	**100**	178.6 ± 71.1	477.1 ± 120.3	83	84	116.2 ± 386.3	294.2 ± 131.4
HiPBI ($h = 50$, A)	73	**100**	40.1 ± 76.9	324.3 ± 169.7	85	87	100.0 ± 357.9	**293.4 ± 123.7**
HiPBI ($h = 75$, A)	**100**	**100**	**8.5 ± 6.0**	205.8 ± 35.3	86	87	106.1 ± 376.5	297.3 ± 122.1
HiPBOT ($h = 5$, A)	0	**100**	199.1 ± 1.1	500.0 ± 0.0	94	94	55.1 ± 170.5	321.9 ± 176.2
HiPBOT ($h = 10$, A)	**100**	**100**	22.2 ± 4.2	147.9 ± 20.3	**99**	**99**	**20.5 ± 99.6**	286.0 ± 80.9
HiPBOT ($h = 15$, A)	**100**	**100**	17.5 ± 3.3	**126.4 ± 18.2**	94	94	59.8 ± 203.4	330.6 ± 188.3

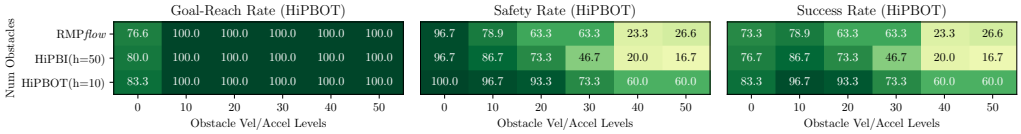

Figure 3: Comparative evaluation for different velocity and acceleration levels of obstacles (30 seeds per evaluation).

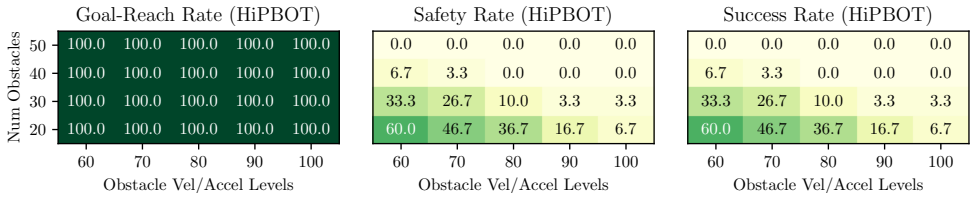

Figure 4: Stress test of HiPBOT on extreme velocity and noisy acceleration levels. We run 30 seeds per evaluation.

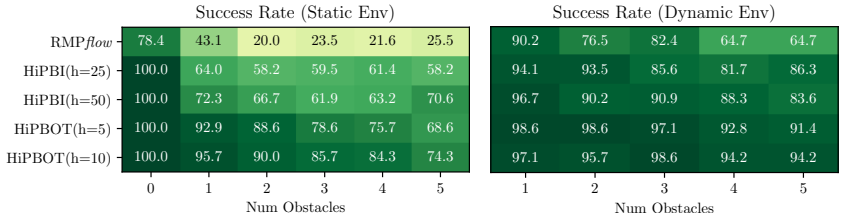

Figure 5: Comparative study in the manipulation environment in static and dynamic obstacles, with increasing obstacle number. We run 70 seeds per evaluation.

4.3. Whole-Body Environment

Finally, we demonstrate HiPBOT capabilities in the MEMA setting with a high-dimensional, multi-objective and highly dynamic environment, see Fig. 1. As demonstrated in the videos, HiBPOT is

able to compromise between objectives thanks to its ability to adapt expert priorities online. In contrast, RMPflow struggles to find good situational actions and eventually collides.

5. Related work

Reactive motion generation. Groundbreaking work was realized with OSC by Khatib (1987), that introduced artificial potential fields for modeling obstacle avoidance (repulsive) and goal-reaching (attractive) behaviors. RMPs (Ratliff et al., 2018; Cheng et al., 2018; Xie et al., 2020) consider geodesics in the vicinity of obstacles using Riemannian metrics. Learning-based methods learn reactive and stable primitives (Khansari-Zadeh and Billard, 2011; Ijspeert et al., 2013; Calinon et al., 2014). Blending of primitives were introduced in (Luksch et al., 2012; Saveriano et al., 2019), in probabilistic settings (Paraschos et al., 2018), and in QP-optimization (Jaquier et al., 2022). Recently, Hansel et al. (2022) proposed a blending as inference approach for adjusting the weights of a product of experts. Blending also emerges from cost-function formulations as energy-based models (Lambert et al., 2022; Urain et al., 2022). Fast obstacle avoidance relies on fast perception (Huber et al., 2022), and can be realized via safe learning (Liu et al., 2022b,a). While object-centric primitives are locally reactive, they tend to get stuck in local minima, as they lack look-ahead capabilities.
Hierarchical planning and control. These approaches refer to multi-level planners or operate in the parameter space of motion policies. The former, such as task and motion planning (TAMP), hierarchical planning, or hierarchical reinforcement learning (HRL) (Kaelbling and Lozano-Pérez, 2010; Srivastava et al., 2014; Pertsch et al., 2020), generate sub-goals that an underlying planner or policy must reach. Methods in HRL either adjust constraint functions of dynamic motion primitives (Bahl et al., 2020, 2021) or select a policy from a mixture of experts (Daniel et al., 2012; End et al., 2017; Akrour et al., 2021; Zaki et al., 2022). Hierarchical mixture of experts selects only one of the experts (Celik et al., 2022) to act in the environment. In the case of unexpected environmental changes, this selective behavior leads to sub-optimal performance (Kroemer et al., 2015). We compose simple and stable reactive policies that lead to complex reactive robot behaviors.
Optimal transport in robot planning. While OT has several practical applications in resource assignment problems and machine learning (Peyré et al., 2019), its application to robotics is scarce. Most applications consider swarm and multi-robot coordination (Inoue et al., 2021; Krishnan and Martínez, 2018; Bandyopadhyay et al., 2014; Kabir and Lee, 2021; Frederick et al., 2022), while OT can be used for exploring while planning (Kabir and Lee, 2020), for imitation learning (Haldar et al., 2023), and for curriculum learning (Klink et al., 2022). A comprehensive review of OT in control is available in (Chen et al., 2021). To the best of our knowledge, we are the first to introduce UOT-based inference of reactive robot planning and policy blending.

6. Conclusion

We proposed an efficient hierarchical policy blending framework as entropic-regularized UOT, where the upper level evaluates the expert contributions based on the task costs and adapts the weight scaling of lower-level experts-agents. Our method introduces an efficient look-ahead evaluation into myopic composable frameworks, thereby improving their reactiveness, safety guarantees, and has better chances of avoiding local minima. There are multiple exciting directions to be explored, e.g., better exploration mechanisms in HiPBOT, as the current cost evaluation only allows as many rollouts as the number of experts. Further, casting the policy blending problem as OT opens a novel view about interactions of multi-experts to multi-agents, which is largely under-explored.

Acknowledgments

This research is supported by the German Research Foundation through the collaborative projects METRIC4IMITATION (PE 2315/11-1) CHIRON, and the Emmy Noether Programme (CH 2676/1-1). The authors would like to thank Joao Carvalho for his feedback, and Snehal Jauhri for his advice on setting up the whole-body control environment.

References

Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows. *Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts*, 1988.

Riad Akrour, Davide Tateo, and Jan Peters. Continuous action reinforcement learning from a mixture of interpretable experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems*, 30, 2017.

Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. *Advances in Neural Information Processing Systems*, 33: 5058–5069, 2020.

Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Hierarchical neural dynamic policies. In *Robotics: Science and Systems*, 2021.

Saptarshi Bandyopadhyay, Soon-Jo Chung, and Fred Y Hadaegh. Probabilistic swarm guidance using optimal transport. In *2014 IEEE Conference on Control Applications (CCA)*, pages 498–505. IEEE, 2014.

Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 750–759. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/bhardwaj22a.html.

Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, volume 49. Springer, 2019.

Sylvain Calinon, Danilo Bruno, and Darwin G Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3344. IEEE, 2014.

Onur Celik, Dongzhuoran Zhou, Ge Li, Philipp Becker, and Gerhard Neumann. Specializing versatile skill libraries using local mixture of experts. In *Conference on Robot Learning*, pages 1423–1433. PMLR, 2022.

Yongxin Chen, Tryphon T Georgiou, and Michele Pavon. Optimal transport in systems and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 2021.

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*, pages 441–457. Springer, 2018.

Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. Rmpflow: A geometric framework for generation of multitask motion policies. *IEEE Transactions on Automation Science and Engineering*, 18(3):968–987, 2021.

Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

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

Felix End, Riad Akrour, Jan Peters, and Gerhard Neumann. Layered direct policy search for learning hierarchical skills. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6442–6448. IEEE, 2017.

Fabrizio Flacco, Alessandro De Luca, and Oussama Khatib. Control of redundant robots under hard joint constraints: Saturation in the null space. *IEEE Transactions on Robotics*, 31(3):637–654, 2015.

Christina Frederick, Magnus Egerstedt, and Haomin Zhou. Collective motion planning for a group of robots using intermittent diffusion. *Journal of Scientific Computing*, 90(1):1–20, 2022.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. *Advances in neural information processing systems*, 28, 2015.

Siddhant Haldar, Vaibhav Mathur, Denis Yarats, and Lerrel Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *Conference on Robot Learning*, pages 32–43. PMLR, 2023.

Kay Hansel, Julen Urain, Jan Peters, and Georgia Chalvatzaki. Hierarchical policy blending as inference for reactive robot control. *arXiv preprint arXiv:2210.07890*, 2022.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002.

Neville Hogan and Dagmar Sternad. Dynamic primitives of motor behavior. *Biological cybernetics*, 106(11):727–739, 2012.

Lukas Huber, Aude Billard, and Jean-Jacques Slotine. Fast obstacle avoidance based on real-time sensing. *arXiv preprint arXiv:2205.04928*, 2022.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2): 328–373, 2013.

Daisuke Inoue, Yuji Ito, and Hiroaki Yoshida. Optimal transport-based coverage control for swarm robot systems: Generalization of the voronoi tessellation-based method. In *2021 American Control Conference (ACC)*, pages 3032–3037. IEEE, 2021.

Noémie Jaquier, You Zhou, Julia Starke, and Tamim Asfour. Learning to sequence and blend robot skills via differentiable optimization. *arXiv preprint arXiv:2206.00559*, 2022.

Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Robot learning of mobile manipulation with reachability behavior priors. *IEEE Robotics and Automation Letters*, 7(3):8399–8406, 2022. doi: 10.1109/LRA.2022.3188109.

Rabiul Hasan Kabir and Kooktae Lee. Receding-horizon ergodic exploration planning using optimal transport theory. In *2020 American Control Conference (ACC)*, pages 1447–1452. IEEE, 2020.

Rabiul Hasan Kabir and Kooktae Lee. Efficient, decentralized, and collaborative multi-robot exploration using optimal transport theory. In *2021 American Control Conference (ACC)*, pages 4203–4208, 2021. doi: 10.23919/ACC50511.2021.9483227.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011. doi: 10.1109/ICRA.2011.5980280.

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.

S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011. doi: 10.1109/TRO.2011.2159412.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

Oussama Khatib, Luis Sentis, Jaeheung Park, and James Warren. Whole-body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, 1(01):29–43, 2004.

Pascal Klink, Haoyi Yang, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. Curriculum reinforcement learning via constrained optimal transport. In *International Conference on Machine Learning*, pages 11341–11358. PMLR, 2022.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *IJRR*, 2013.

Vishaal Krishnan and Sonia Martínez. Distributed optimal transport for the deployment of swarms. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4583–4588. IEEE, 2018.

Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke Van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1503–1510. IEEE, 2015.

Alexander Lambert, An T Le, Julen Urain, Georgia Chalvatzaki, Byron Boots, and Jan Peters. Learning implicit priors for motion optimization. *arXiv preprint arXiv:2204.05369*, 2022.

Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. *Computer Science Dept. Oct.*, 98(11), 1998.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Zhiyuan Hu, Jan Peters, and Georgia Chalvatzaki. Safe reinforcement learning of dynamic high-dimensional robotic tasks: navigation, manipulation, interaction. *arXiv preprint arXiv:2209.13308*, 2022a.

Puze Liu, Kuo Zhang, Davide Tateo, Snehal Jauhri, Jan Peters, and Georgia Chalvatzaki. Regularized deep signed distance fields for reactive motion generation. *arXiv preprint arXiv:2203.04739*, 2022b.

Tobias Luksch, Michael Gienger, Manuel Mühlig, and Takahide Yoshiike. Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2082–2088. IEEE, 2012.

Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018. doi: 10.1177/0278364918790369.

James Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the Twentieth annual ACM symposium on Theory of Computing*, pages 377–387, 1988.

Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42(3):529–551, 2018.

Dae-Hyung Park, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 91–98. IEEE, 2008.

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Shenghao Zhou, Dinesh Jayaraman, Chelsea Finn, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *Advances in Neural Information Processing Systems*, 33:17321–17333, 2020.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Khiem Pham, Khang Le, Nhat Ho, Tung Pham, and Hung Bui. On unbalanced optimal transport: An analysis of sinkhorn algorithm. In *International Conference on Machine Learning*, pages 7673–7682. PMLR, 2020.

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

Matteo Saveriano, Felix Franzel, and Dongheui Lee. Merging position and orientation motion primitives. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7041–7047. IEEE, 2019.

Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021.

Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.

Julen Urain, Niklas Funk, Georgia Chalvatzaki, and Jan Peters. Se (3)-diffusionfields: Learning cost functions for joint grasp and motion optimization through diffusion. *arXiv preprint arXiv:2209.03855*, 2022.

Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. *arXiv preprint arXiv:2008.07792*, 2020.

Mandy Xie, Karl Van Wyk, Anqi Li, Muhammad Asif Rana, Qian Wan, Dieter Fox, Byron Boots, and Nathan Ratliff. Geometric fabrics for the acceleration-based design of robotic motion. *arXiv preprint arXiv:2010.14750*, 2020.

Mohammadi Zaki, Avi Mohan, Aditya Gopalan, and Shie Mannor. Actor-critic based improper reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25867–25919. PMLR, 17–23 Jul 2022.

Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10): 1164–1193, 2013. doi: 10.1177/0278364913488805.