Towards safe and efficient learning in the wild: guiding RL with constrained uncertainty-aware movement primitives

Abhishek Padalkar[1], Freek Stulp[1], Gerhard Neumann[2], João Silvério[1]

***Abstract*—Guided Reinforcement Learning (RL) presents an effective approach for robots to acquire skills efficiently, directly in real-world environments. Recent works suggest that incorporating hard constraints into RL can expedite the learning of manipulation tasks, enhance safety, and reduce the complexity of the reward function. In parallel, learning from demonstration (LfD) using movement primitives is a well-established method for initializing RL policies. In this paper, we propose a constrained, uncertainty-aware movement primitive representation that leverages both demonstrations and hard constraints to guide RL. By incorporating hard constraints, our approach aims to facilitate safer and sample-efficient learning, as the robot need not violate these constraints during the learning process. At the same time, demonstrations are employed to offer a baseline policy that supports exploration. Our method improves state-of-the-art techniques by introducing a projector that enables state-dependent noise derived from demonstrations while ensuring that the constraints are respected throughout training. Collectively, these elements contribute to safe and efficient learning alongside streamlined reward function design. We validate our framework through an insertion task involving a torque-controlled, 7-DoF robotic manipulator.**

***Index Terms* – Safe Reinforcement Learning, Learning from Demonstrations, Constrained Learning, Guided Reinforcement Learning**

I. Introduction

Learning from Demonstration (LfD) [1] has proven to be an effective method for motion generation, enabling a robot to imitate and adapt demonstrated motions. Various frameworks have been developed, including Dynamic Movement Primitives (DMPs) [2], Probabilistic Movement Primitives (ProMPs) [3], and Kernelized Movement Primitives (KMPs) [4], which effectively address common real-world challenges such as generalizing to new situations and avoiding obstacles. However, these methods often struggle in dynamic environments where demonstrations inadequately represent task dynamics, particularly during in-contact tasks. In such tasks, the policy often receives out-of-distribution states as an input, resulting in failures. Collaborative robots aim to mitigate these challenges by employing impedance control to remain compliant while in contact, thus reacting to the inaccuracies caused by kinematics and dynamics. However, learning a robust LfD policy that can adapt to such uncertainties remains a significant challenge.

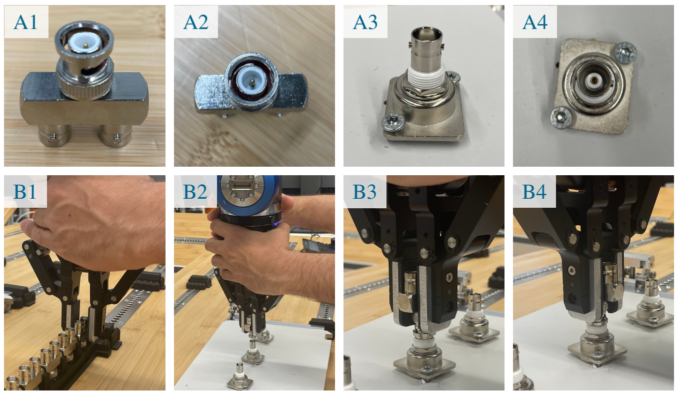


Fig. 1. BNC connector assembly task from the NIST assembly benchmark [5]. A1 to A4 show the male and female BNC connectors. B1 to B4 show a human demonstrating the task by hand-guiding the DLR SARA robot [6]. An LfD trajectory learned from the demonstrations was not able to solve the task as it does not model the contact dynamics and uncertainties in the kinematics.

Reinforcement Learning (RL) addresses this challenge by training a reactive policy that considers the current state of both the robot and its environment. However, the necessity of a large number of trials, coupled with concerns about robot safety, presents a significant challenge for RL to be directly applicable on real robots. Transfer learning attempts to overcome this by learning a policy in a simulation and then applying it to the robot, yet it is still limited by the sim-to-real gap [7]. Learning skills directly on real robots eliminates the need for meticulously modelling tasks in simulation. In [8] we propose a guided RL approach enabling tasks to be learned directly on the real robot, where available task knowledge, represented as constraints, facilitates effective policy search (see Section II for an overview of related work). Despite the promising results, the manual modeling of inductive biases (e.g. constraints, exploration strategies) can be challenging, particularly in complex tasks that involve contacts.

To address the above-mentioned challenges, we propose to learn a nominal policy together with a state-dependent exploration strategy from human demonstrations. Specifically, we introduce a novel movement primitive representation, Linearly Constrained Null-Space Kernelized Movement Primitives (LC-NS-KMP), where a non-parametric LfD framework generates motions while adhering to linear inequality constraints on the robot state. Simultaneously, LC-NS-KMP provides a *null-space* projector that allows actions generated by RL policies to modify the mean behavior of the LfD policy. The derived projector modifies the mean behavior of the LfD policy in accordance with the variance and correlations in the demon-

Manuscript received: December 9, 2024; Revised: March 10, 2025; Accepted: April 13, 2025.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments.

This work was partially funded by the DLR project "Factory of the Future Extended" and the European Union's Horizon Research and Innovation Programme under Grant 101136067 (INVERSE).

[1]German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchener Str. 20, 82234 Weßling, Germany, `firstname.lastname@dlr.de`

[2]Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany, `firstname.lastname@kit.edu`

Digital Object Identifier (DOI): see top of this page.

strations. Consequently, the same null-space action will result in larger/smaller deviations in states where the variance in the demonstrations is higher/lower. This uncertainty-awareness facilitates the design of state-dependent exploration noise. We leverage this property to enable state-based exploration in RL while ensuring safety by enforcing state space constraints. The main contribution of this paper is thus a new RL framework, *Kernelized Guided Reinforcement Learning (KGRL)*, described in Section III, which leverages the properties of LC-NS-KMP to facilitate RL in the wild.

To fully demonstrate the capabilities of KGRL, we selected the BNC connector assembly task from the NIST assembly task board 1 [5], illustrated in Fig. 1. This task presents significant challenges, as it requires precise insertion of the connector while maintaining compliance to prevent damage to the components. Following the insertion, a complex series of translations and rotations are necessary to lock the connector in position. Our method is well-suited for such tasks because it 1) allows for the specification of constraints that ensure safe operation during state space exploration, and 2) guarantees uncertainty-aware, state-dependent exploration for reinforcement learning, which helps avoid unnecessary exploration in the low-variance regions of the motion (Section IV).

II. Related Work

Learning from Demonstration is a widely used approach for learning robot motions from humans. Ravichandar *et al.* [1] present a comprehensive survey on recent advances in LfD. One of the main paradigms in LfD is behavior cloning (BC) which uses supervised learning frameworks such as Dynamic Motion Primitives (DMPs) [2], Probabilistic Movement Primitives (ProMPs) [3] and Kernelized Movement Primitives (KMPs) [4] to teach new skills to robots with only a few demonstrations. BC often generates brittle policies that fail when the robot encounters situations outside the distribution of the demonstrations [9]. Another important paradigm is Inverse Reinforcement Learning (IRL) where a reward function is extracted from human demonstrations to guide RL. Extracting reward functions in IRL is also subject to the coverage of optimal behaviors in the used demonstrations [10], [11].

The limitations of LfD, particularly in tasks involving contacts, can be mitigated by the use of RL in combination with demonstrations, to obtain more robust policies [12] than with LfD alone. This has been done in four main ways: (1) by populating the replay buffer of off-policy RL algorithms with demonstrations and associated reward [13], [14], (2) by extracting a reward function from demonstrations and augmenting it with task specific rewards for further generalization while simultaneously learning an RL policy (GAIL [11], AIRL [15]), (3) by using a behavior cloning policy to regularize the RL policy during learning [16], [17], and (4) by learning a residual RL policy to support an LfD one [18], [19]. Populating the replay buffer with demonstrations provides initial experience from the demonstrations, but the RL policy is initialized randomly and trained from scratch which is an inefficient use of the demonstrations. Learning a reward function and an RL policy simultaneously from the demonstrations can result in unstable learning [9].

Work presented in [20] leverages trajectories produced by a trajectory-optimization-based controller to initialize an RL policy and then learns robust behaviors with RL in simulation using domain randomization. They propose to learn adaptive task phase dynamics to facilitate learning policies which are robust against failures. Work presented in [19] proposes multiple strategies for correcting a DMP policy with RL residual policy for solving contact-rich tasks. None of the above methods extract exploration strategies from demonstrations or account for state-dependent noise, instead often assuming isotropic exploration noise. Moreover, in these approaches, the exploration is unconstrained, hence potentially unsafe exploratory actions can cause damage to the environment as well as the robot. Both of these shortcomings are addressed in our proposed solution.

A holistic review of works on safe reinforcement learning is presented in [21]. Approaches such as [22] and [23] present an optimization layer to optimize the actions generated by the RL policy based on safety constraints. Others, e.g. [24], [25] use Gaussian Processes for conducting safe exploration in the proximity of already safe states. Our approach has similarities with these works as we propose to introduce modifications in the trajectories based on variance in the already safe demonstrations. Additionally, we provide a mechanism to enforce hard constraints on the state of the robot.

More recently, Chi *et al.* [26] introduced diffusion policies for LfD which leverage a conditional denoising diffusion process for generating robot motions. An overview of diffusion policies for RL is provided in [27]. Zheng et al. introduce diffusion-based, safe RL [28] by defining a feasibility-dependent objective, which depends on an offline dataset to predefine a safe region. To the best of our knowledge, current diffusion-based methods in robot manipulation lack inherent constraint enforcement mechanisms and uncertainty-aware exploration.

Kernelized Movement Primitives (KMPs) are an LfD framework originally formulated by Huang *et al.* [4]. Silvério and Huang [29] extended this framework such that modulations in the trajectories are possible with a *null-space* modifier, without re-parameterizing the whole KMP. Later, Huang and Caldwell [30] introduced a way of enforcing linear constraints on the predicted trajectories without re-parameterization. In our work, we build on top of the above-mentioned works by deriving a unified framework for enforcing linear inequality constraints and modifying trajectories with a null-space action. We leverage the resulting framework in combination with RL, to efficiently and safely learn in-contact tasks bootstrapped by human demonstrations.

III. Methodology

A. Background

a) Kernelized movement primitives (KMP): Huang *et al.* [4] presented an approach to learn probabilistic trajectories from demonstrations called Kernelized Movement Primitives (KMP). Consider M demonstrations $D = \{\{\boldsymbol{s}_{n,m}, \boldsymbol{\eta}_{n,m}\}_{n=1}^{N}\}_{m=1}^{M}$ where N is the length of a trajectory comprised of input $\boldsymbol{s} \in \mathbb{R}^{\mathcal{I}}$ and corresponding output

$\boldsymbol{\eta} \in \mathbb{R}^{\mathcal{O}}$. A probabilistic policy can be learned from these demonstrations using Gaussian Mixture Model (GMM) such that, $\mathcal{P}(s, \boldsymbol{\eta}) \sim \sum_{c=1}^{C} p_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, where, p_c, $\boldsymbol{\mu}_c$,and $\boldsymbol{\Sigma}_c$ are the prior probability, mean and variance of the c^{th} Gaussian. We can employ Gaussian Mixture Regression (GMR) to obtain a reference trajectory $T_r = \{\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^{N}$ from the learned GMM, where $\hat{\boldsymbol{\mu}}_n$ and $\hat{\boldsymbol{\Sigma}}_n$ are means and covariance matrices computed at each new input. At the same time, a parametric trajectory can also be learned from the same demonstrations,

$$\boldsymbol{\eta}(\boldsymbol{s}) = \boldsymbol{\Theta}(\boldsymbol{s})^{\top}\mathbf{w}, \quad \boldsymbol{\Theta}(\boldsymbol{s}) = \begin{bmatrix} \boldsymbol{\varphi}(\boldsymbol{s}) & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\varphi}(\boldsymbol{s}) & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \boldsymbol{\varphi}(\boldsymbol{s}) \end{bmatrix}, \quad (1)$$

where the matrix $\boldsymbol{\Theta} \in \mathbb{R}^{\mathcal{BO}\times\mathcal{O}}$, weight vector $\mathbf{w} \in \mathbb{R}^{\mathcal{BO}}$, with $\boldsymbol{\varphi}(\boldsymbol{s})$ being a $\mathcal{B}$-dimensional basis function. Consider weights $\mathbf{w}$ are drawn from $\mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, hence we can write $\boldsymbol{\eta}(\boldsymbol{s}) \sim \mathcal{N}(\boldsymbol{\Theta}(\boldsymbol{s})^{\top}\boldsymbol{\mu}_w, \boldsymbol{\Theta}(\boldsymbol{s})^{\top}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\boldsymbol{s}))$. The original formulation [4] proposes to minimize the KL-divergence between the above-mentioned two Gaussian distributions, represented by T_r and $\mathcal{N}(\boldsymbol{\Theta}(\boldsymbol{s})^{\top}\boldsymbol{\mu}_w, \boldsymbol{\Theta}(\boldsymbol{s})^{\top}\boldsymbol{\Sigma}_w\boldsymbol{\Theta}(\boldsymbol{s}))$, leading to a *mean minimization subproblem* with cost function

$$\underset{\boldsymbol{\mu}_w}{\textbf{argmin}} \quad \sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\top}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n) + \frac{1}{2}\lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w. \quad (2)$$

The solution of (2) leads to the formulation of KMPs. The prediction of a KMP is given by $\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s})) = \boldsymbol{k}^*(\boldsymbol{K} + \lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu}$, where $\boldsymbol{\mu} = [\hat{\boldsymbol{\mu}}_1^{\top}, \hat{\boldsymbol{\mu}}_2^{\top}, \dots, \hat{\boldsymbol{\mu}}_N^{\top}]^{\top}$, $\boldsymbol{\Sigma} = \text{blockdiag}(\hat{\boldsymbol{\Sigma}}_1, \hat{\boldsymbol{\Sigma}}_2, \dots, \hat{\boldsymbol{\Sigma}}_N)$, $\boldsymbol{k}^*$ and $\boldsymbol{K}$ are kernel matrices obtained after applying kernel treatment to the basis functions, which will be discussed in detail in Section III-B, and λ is a regularization term. We here only focus on the mean minimization subproblem as our goal is to extract a policy for the robot to track.

b) Linearly-constrained KMP: Huang and Caldwell [30] formulated a linearly constrained imitation learning framework which incorporates linear inequality constraints on the state of the robot, and applied the same method to minimize the KL-divergence between two distributions as [4], to obtain a *constrained* mean minimization subproblem

$$\begin{aligned} \underset{\boldsymbol{\mu}_w}{\textbf{argmin}} \quad & \sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\top}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n) \\ & + \frac{1}{2}\lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w \\ \textbf{s.t.} \quad & \boldsymbol{g}_{n,f}^{\top}\boldsymbol{\eta}(s_n) \geq c_{n,f}, \forall f \in \{1, 2, \dots, F\}, \\ & \forall n \in \{1, 2, \dots, N\}, \end{aligned} \quad (3)$$

where F is the number of constraints imposed on the output and $\boldsymbol{g}_{n,f}$ and $c_{n,f}$ parameterize the constraint hyperplanes. The mean prediction of LC-KMP is given by

$$\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}^*)) = \boldsymbol{k}^*(\boldsymbol{K} + \lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu} + \boldsymbol{k}^*(\boldsymbol{K} + \lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha}, \quad (4)$$

where,

$$\begin{aligned} \boldsymbol{G}_n &= [\boldsymbol{g}_{n,1}\ \boldsymbol{g}_{n,2}\ \boldsymbol{g}_{n,3}\ \dots\ \boldsymbol{g}_{n,F}], \forall n \in \{1, 2, 3, \dots, N\}, \\ \bar{\boldsymbol{G}} &= \text{blockdiag}(\boldsymbol{G}_1,\ \boldsymbol{G}_2,\ \boldsymbol{G}_3,\ \dots,\ \boldsymbol{G}_N), \\ \boldsymbol{\alpha} &= [\alpha_{1,1},\ \alpha_{1,2},\ \dots,\ \alpha_{1,F},\ \dots,\ \alpha_{N,1},\ \dots,\ \alpha_{N,F}]. \end{aligned}$$

The Lagrange multiplier vector $\boldsymbol{\alpha}$ is obtained by solving a convex optimization problem [30]. Note that the prediction given by Eq. (4) respects the constraints defined in Eq. (3).

B. LC-NS-KMP formulation

In this paper, we derive a unified method which combines null-space modifier for KMPs proposed by [29] and linear constraints proposed by [30]. Combining the desirable properties of these methods, our framework allows RL to modulate a mean trajectory predicted by KMPs adhering to linear constraints and covariance in the demonstrations. It helps RL conduct an effective search by modulating the exploration noise in accordance with the variance and constraints.

We start from the same constrained mean optimization problem as Eq. (3), and introduce an additional cost term $\frac{1}{2}\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)$ which results in a *soft* null space projector that modifies the mean trajectory (see [29] for details), to obtain,

$$\begin{aligned} \underset{\boldsymbol{\mu}_w}{\textbf{argmin}} \quad & \sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\top}\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n) \\ & + \frac{1}{2}\lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w + \frac{1}{2}\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w), \\ \textbf{s.t.} \quad & \boldsymbol{g}_{n,f}^{\top}\boldsymbol{\eta}(s_n) \geq c_{n,f}, \forall f \in \{1, 2, \dots, F\}, \\ & \forall n \in \{1, 2, \dots, N\}. \end{aligned} \quad (5)$$

The term $\frac{1}{2}\lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w$ regularizes the solution and the cost term $\frac{1}{2}\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)$ inspired from [29] keeps the solution close to a desired one $\hat{\boldsymbol{\mu}}_w$. Similarly to [30], we propose to solve Eq. (3) by introducing Lagrange multipliers $\alpha_{n,f} \geq 0$, with the Lagrange function

$$\begin{aligned} L(\boldsymbol{\mu}_w, \alpha) = & \sum_{n=1}^{N} \frac{1}{2}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_n)^{\top}\hat{\boldsymbol{\Sigma}}_n^{-1}(\boldsymbol{\Theta}^{\top}(s_n)\boldsymbol{\mu}_w \\ & - \hat{\boldsymbol{\mu}}_n) + \frac{1}{2}\lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w + \frac{1}{2}\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w) \\ & - \sum_{n=1}^{N}\sum_{f=1}^{F} \alpha_{n,f}(\mathbf{g}_{n,f}^{\top}\boldsymbol{\Theta}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w - c_{n,f}), \end{aligned} \quad (6)$$

which can be re-written using matrix notation as

$$\begin{aligned} L(\boldsymbol{\mu}_w, \boldsymbol{\alpha}) = & \frac{1}{2}(\boldsymbol{\Phi}^{\top}\boldsymbol{\mu}_w - \boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\Phi}^{\top}\boldsymbol{\mu}_w - \boldsymbol{\mu}) \\ & + \frac{1}{2}\lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w + \frac{1}{2}\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w)^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{\mu}}_w) \\ & - \boldsymbol{\alpha}^{\top}\bar{\boldsymbol{G}}^{\top}\boldsymbol{\Phi}^{\top}\boldsymbol{\mu}_w - \boldsymbol{\alpha}^{\top}\bar{\boldsymbol{C}}, \end{aligned} \quad (7)$$

where $\boldsymbol{\Phi} = [\boldsymbol{\Theta}(\boldsymbol{s}_1)\ \ \boldsymbol{\Theta}(\boldsymbol{s}_2)\ \ \dots\ \ \boldsymbol{\Theta}(\boldsymbol{s}_N)]$, and $\bar{\boldsymbol{C}} = [\boldsymbol{C}_1^{\top}\ \ \boldsymbol{C}_2^{\top}\ \ \dots\ \ \boldsymbol{C}_N^{\top}]^{\top}$ with $\boldsymbol{C}_n = [c_{n,1}\ c_{n,2}\ \dots\ c_{n,F}]^{\top}, \forall n \in \{1, 2, \dots, N\}$. By setting the derivative $\frac{\partial L(\boldsymbol{\mu}_w, \boldsymbol{\alpha})}{\partial \boldsymbol{\mu}_w} = 0$, we obtain

$(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \gamma\boldsymbol{I})\boldsymbol{\mu}_w^* = (\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \beta\hat{\boldsymbol{\mu}}_w + \boldsymbol{\Phi}\bar{\boldsymbol{G}}\boldsymbol{\alpha})$ resulting in

$$\begin{aligned}\boldsymbol{\mu}_w^* &= (\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \gamma\boldsymbol{I})^{-1}(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \beta\hat{\boldsymbol{\mu}}_w + \boldsymbol{\Phi}\bar{\boldsymbol{G}}\boldsymbol{\alpha}), \quad (8)\\ &= \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}(\boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Phi}^\top)\hat{\boldsymbol{\mu}}_w, \quad (9)\end{aligned}$$

where, $\boldsymbol{A} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \gamma\boldsymbol{\Sigma})^{-1}$ and $\gamma = \lambda + \beta$. Equation (8) is further simplified into Eq. (9) using the Woodbury identity[1]. By substituting $\boldsymbol{\mu}_w^*$ in Eq. (7) and Eq. (1), we get

$$\begin{aligned}\tilde{L}(\boldsymbol{\alpha}) &= \boldsymbol{\alpha}^\top\bar{\boldsymbol{G}}^\top\boldsymbol{\Sigma}\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + (2\boldsymbol{\mu}^\top\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\\ &\quad - \beta\hat{\boldsymbol{\mu}}_w^\top\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top)\boldsymbol{\alpha} + \text{const}, \quad (10)\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}^*)) &= \boldsymbol{\Theta}(\boldsymbol{s}^*)\boldsymbol{\mu}_w^*\\ &= \boldsymbol{\Theta}(\boldsymbol{s}^*)(\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}(\boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Phi}^\top)\hat{\boldsymbol{\mu}}_w), \quad (11)\end{aligned}$$

respectively, where $\mathcal{A} = -\frac{1}{2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} - \frac{\gamma}{2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$.

For a desired output $\boldsymbol{\xi} = \hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\mu}}_w$, we can estimate the optimal weight vector $\hat{\boldsymbol{\mu}}_w$ given the target trajectory $\boldsymbol{\xi}$, using the right pseudo-inverse of $\hat{\boldsymbol{\Phi}}^\top$, similarly to [29], hence, $\hat{\boldsymbol{\mu}}_w = \hat{\boldsymbol{\Phi}}(\hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}})^{-1}\boldsymbol{\xi}$. By further replacing $\hat{\boldsymbol{\mu}}_w$ in Eq. (10) and Eq. (11), we obtain, respectively,

$$\begin{aligned}\tilde{L}(\boldsymbol{\alpha}) &= \boldsymbol{\alpha}^\top\bar{\boldsymbol{G}}^\top\boldsymbol{\Sigma}\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + (2\boldsymbol{\mu}^\top\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\\ &\quad - \beta\boldsymbol{\xi}^\top(\hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}})^{-1}\hat{\boldsymbol{\Phi}}^\top\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top)\boldsymbol{\alpha} + \text{const},\end{aligned} \quad (12)$$

and

$$\begin{aligned}\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}^*)) &= \boldsymbol{\Theta}(\boldsymbol{s}^*)(\boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha}\\ &\quad + \frac{\beta}{\gamma}(\boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{A}\boldsymbol{\Phi}^\top)\hat{\boldsymbol{\Phi}}(\hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}})^{-1}\boldsymbol{\xi}).\end{aligned} \quad (13)$$

Similarly to [4], we propose to kernelize the above equation using the kernel treatment, i.e. the inner product of basis functions $\varphi(\boldsymbol{s}_i)$ and $\varphi(\boldsymbol{s}_j)$ defined as $\varphi(\boldsymbol{s}_i)^\top\varphi(\boldsymbol{s}_j) = k(\boldsymbol{s}_i, \boldsymbol{s}_j)$, where $k(.,.)$ is a kernel function. With the kernel treatment, we can write

$$\begin{aligned}\tilde{L}(\boldsymbol{\alpha}) &= \boldsymbol{\alpha}^\top\bar{\boldsymbol{G}}^\top\boldsymbol{\Sigma}\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + (2\boldsymbol{\mu}^\top\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\\ &\quad - \beta\boldsymbol{\xi}^\top\underline{\boldsymbol{K}}^{-1}\hat{\boldsymbol{K}}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top)\boldsymbol{\alpha} + \text{const}, \quad (14)\end{aligned}$$

$$\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}^*)) = \boldsymbol{k}^*\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{k}^*\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^*\boldsymbol{A}\hat{\boldsymbol{K}})\underline{\boldsymbol{K}}^{-1}\boldsymbol{\xi}, \quad (15)$$

with $\boldsymbol{A} = (\boldsymbol{K} + \lambda\boldsymbol{\Sigma})^{-1}$, and $\mathcal{A} = -\frac{1}{2}\boldsymbol{K}\boldsymbol{\Sigma}^{-1}\boldsymbol{K} - \frac{\gamma}{2}\boldsymbol{K}$, where,

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{s}_1, \boldsymbol{s}_1) & \dots & \boldsymbol{k}(\boldsymbol{s}_1, \boldsymbol{s}_N) \\ \vdots & \ddots & \vdots \\ \boldsymbol{k}(\boldsymbol{s}_N, \boldsymbol{s}_1) & \dots & \boldsymbol{k}(\boldsymbol{s}_N, \boldsymbol{s}_N)) \end{bmatrix},$$

$$\boldsymbol{k}^* = [\boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}_1), \dots, \boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}_N)], \qquad \boldsymbol{k}(\boldsymbol{s}_i, \boldsymbol{s}_j) = k(\boldsymbol{s}_i, \boldsymbol{s}_j)\boldsymbol{I},$$
$$\underline{\boldsymbol{K}} = \hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}}, \qquad \hat{\boldsymbol{K}} = \boldsymbol{\Phi}^\top\hat{\boldsymbol{\Phi}}, \qquad \hat{\boldsymbol{k}} = \boldsymbol{\Phi}(\boldsymbol{s}^*)^\top\hat{\boldsymbol{\Phi}}.$$

We substitute $\mathcal{B}_1 = \bar{\boldsymbol{G}}^\top\boldsymbol{\Sigma}\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}$, and $\mathcal{B}_2 = 2\boldsymbol{\mu}^\top\boldsymbol{A}\mathcal{A}\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}} + \beta\boldsymbol{\xi}^\top\underline{\boldsymbol{K}}^{-1}\hat{\boldsymbol{K}}(-\boldsymbol{A})\boldsymbol{\Sigma}\bar{\boldsymbol{G}} + \bar{\boldsymbol{C}}^\top$ in Eq. (14), which results in a quadratic function $\tilde{L}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top\mathcal{B}_1\boldsymbol{\alpha} + \mathcal{B}_2\boldsymbol{\alpha}$. Thus, we can tackle the problem of finding optimal Lagrange multipliers $\boldsymbol{\alpha}$ by solving

$$\begin{aligned}\underset{\boldsymbol{\alpha}}{\textbf{argmax}} \quad & \boldsymbol{\alpha}^\top\mathcal{B}_1\boldsymbol{\alpha} + \mathcal{B}_2\boldsymbol{\alpha},\\ \textbf{s.t.} \quad & \boldsymbol{\alpha} \geq 0.\end{aligned} \quad (16)$$

Since $\boldsymbol{A}\mathcal{A}\boldsymbol{A} = (\boldsymbol{A}\mathcal{A}\boldsymbol{A})^\top \preccurlyeq 0$ and $-\boldsymbol{A} = -\boldsymbol{A}^\top \preccurlyeq 0$, Eq. (16) defines a quadratic program with linear inequality constraints. After solving for $\boldsymbol{\alpha}$, Eq. (15) enables constrained predictions incorporating modulations from $\boldsymbol{\xi}$.

C. Properties of LC-NS-KMP

We evaluated the properties of LC-NS-KMP using synthetically-generated 2D time trajectories, shown in Fig. 2 (A1), alongside the learned GMM. We chose the squared exponential kernel $k(t_i, t_j) = \exp(-l(t_i - t_j)^2)$, with hyper-parameter $l = 2$. We use GMR for generating the reference trajectories and covariances, and the same kernel in all our experiments, including those in Section IV. The KMP input is $\boldsymbol{s} = t$ and the 2D outcome is $\boldsymbol{\eta} = [x \quad y]^\top$.

Figure 2 (A2) illustrates the impact of various null-space actions $\boldsymbol{\xi}$ applied at $t = 3.2s$ on the resultant trajectories $\{\boldsymbol{\eta}_i\}_{i=1}^N$. Despite the local modulation in the trajectory, smoothness is preserved while respecting the linear inequality constraints defined in LC-NS-KMP, similar to the approach of [30]. Finally, Fig. 2 (A3) shows trajectories generated using Eq. (15), where $\boldsymbol{\xi}$ is randomly sampled from a normal distribution at each time step. $\boldsymbol{\xi}$ modulates the trajectory in accordance with the variance in the demonstrations and hence it demonstrates the uncertainty-aware exploration through null-space actions. The modulated trajectory also satisfies the constraints despite the noise amplitude. This property paves the way for safe exploration in RL.

D. Kernelized Guided RL (KGRL)

We propose to use null-space actions $\boldsymbol{\xi}$ obtained from an RL policy $\pi(\boldsymbol{\xi}|\boldsymbol{q})$, to introduce modulations in the LfD trajectory learned from the demonstrations[2]. LC-KMP [30] in Eq. (4) predicts a trajectory which respects the linear inequality constraints defined in Eq. (3). Our proposed method LC-NS-KMP in Eq. (15) allows modifications in the prediction using null-space action $\boldsymbol{\xi}$, whose magnitude depends on the variance in the demonstrations, while respecting the constraints in Eq. (5). This important property allows us to conduct efficient and safe RL search using null-space actions. Particularly, we obtain null-space actions from a RL policy $\pi(\boldsymbol{\xi}|\boldsymbol{q})$ modifying the prediction for further refinement as

$$\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}^*)) = \boldsymbol{k}^*\boldsymbol{A}\boldsymbol{\mu} + \boldsymbol{k}^*\boldsymbol{A}\boldsymbol{\Sigma}\bar{\boldsymbol{G}}\boldsymbol{\alpha} + \frac{\beta}{\gamma}(\hat{\boldsymbol{k}}^* - \boldsymbol{k}^*\boldsymbol{A}\hat{\boldsymbol{K}})\pi(\boldsymbol{\xi}|\boldsymbol{q}). \quad (17)$$

Our complete approach, termed Kernelized Guided Reinforcement Learning (KGRL), is summarized in Algorithm 1.

IV. Evaluation

A. Experiments in simulation

We evaluate the performance of our proposed framework against two baselines: 1) a residual RL policy and 2) a

[1]Woodbury identity: if $\boldsymbol{P} \succ 0$ and $\boldsymbol{R} \succ 0$, $(\boldsymbol{P}^{-1} + \boldsymbol{B}^\top\boldsymbol{R}^{-1}\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{R}^{-1} = \boldsymbol{P}\boldsymbol{B}^\top(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^\top + \boldsymbol{R})^{-1}$.

[2]We use $\boldsymbol{q}$ to represent the RL agent state, to distinguish it from the KMP input $\boldsymbol{s}$.

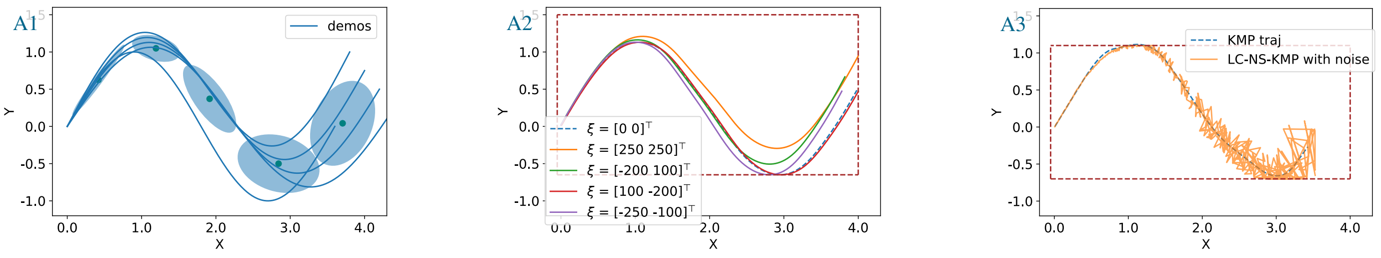


Fig. 2. LC-NS-KMP properties: (A1) shows the demonstrations and the learned GMM; (A2) shows the modulations due to different $\boldsymbol{\xi}$ applied at $t = 3.2s$, adhering to the constraints; (A3) shows the effect of randomly sampled $\boldsymbol{\xi}$. In (A2) and (A3), the modifications introduced by $\boldsymbol{\xi}$ respect the linear inequality constraints, which are shown by dashed red rectangle.

Algorithm 1 Kernelized Guided RL (KGRL)

```
1:  Collect demonstrations $D \leftarrow \{\{\boldsymbol{s}_{n,m}, \boldsymbol{\eta}_{n,m}\}_{n=1}^{N}\}_{m=1}^{M}$
2:  Set $\lambda$, $\beta$ and define $k(.,.)$
3:  Set horizon $h$
4:  Model joint probability distribution $\mathcal{P}(\boldsymbol{s}, \boldsymbol{\eta})$
5:  Define set of constraints $O = \{\{\boldsymbol{g}_{n,f}^{\top}, c_{n,f}\}_{n=1}^{N}\}_{f=1}^{F}$
6:  Initialize a RL policy $\pi(\boldsymbol{\xi}|\boldsymbol{q})$ and policy parameters
7:  loop for each $n = 1, \ldots, N$
8:      In current state $\boldsymbol{s}_n$, retrieve the reference trajectory
        distribution $T_r \leftarrow \{\hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\Sigma}}_i\}_{i=n}^{n+h}$
9:      Choose constraints for $T_r$ from $O$
10:     Sample RL action $\boldsymbol{\xi}$ from RL policy $\pi$
11:     Compute $\boldsymbol{\alpha}$ with Eq. (16)
12:     Compute $\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}_n))$ with Eq. (17)
13:     Apply $\mathbb{E}(\boldsymbol{\eta}(\boldsymbol{s}_n))$ to robot
14:     Compute reward
15:     Update RL policy $\pi$
16: end loop
```

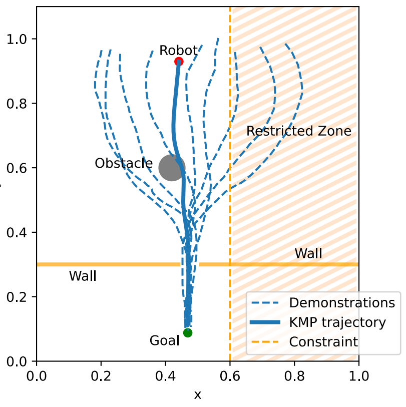


Fig. 3. Simulated 2D environment where the robot navigates through a narrow passage to reach the goal. A trajectory for navigation can be learned from demonstration. Then an RL policy learns to avoid the obstacle in the path of the robot. The robot must not cross into the restricted zone.

safe residual RL policy using predictive safety filters [21]. Both of these residual policies learn to adapt the mean LfD trajectory. For this evaluation, we developed a simulation involving a robot that navigates a 2D environment with the primary objective of reaching a goal position while passing through a narrow passage, see Fig. 3. As illustrated in Fig. 3, demonstrations $D = \{\{t_{n,m}, \boldsymbol{p}_{n,m}\}_{n=1}^{400}\}_{m=1}^{9}$ were given to the robot. However, after demonstrating the trajectories, an obstacle is added along the path of the robot so that the mean trajectory consistently intersects with the obstacle. Moreover, we define a *Restricted Zone* ($x \geq 0.6$) for the robot, as shown in Fig. 3, after the demonstrations were collected. The robot must not go into the Restricted Zone to ensure safety during learning and execution. Consequently, an RL agent must learn to avoid the obstacle and not to enter the restricted zone, while still successfully reaching the goal.

We selected KMP, described in Section III-A, as the baseline LfD method, with $\boldsymbol{s} = t$ and generated the necessary time-based trajectory $\boldsymbol{p}_t^{\text{kmp}} = \mathbb{E}(\boldsymbol{\eta}(t))$ to reach the goal. The residual RL policy $\pi_{\text{res}}(\Delta\boldsymbol{p}_t|t)$ modifies the mean trajectory $\boldsymbol{p}_t^{\text{kmp}}$ for avoiding the obstacle. The robot follows the resultant 2D position $\hat{\boldsymbol{p}}_t = \boldsymbol{p}_t^{\text{kmp}} + \Delta\boldsymbol{p}_t$.

We then implemented predictive safety filters [21] in the form of Active Constraints (AC) similar to our earlier work [8] which keep the robot in the safe zone and avoid wall collisions by filtering the unsafe $\hat{\boldsymbol{p}}_t$ commands. This resulted in the safe residual RL policy $\pi_{\text{safe}}(\Delta\boldsymbol{p}_t|t)$. As illustrated in Fig. 4(b), ACs stop the robot 0.02 units before the restricted zone and the walls. ACs implement projection functions which project an unsafe robot position back to the safe zone before sending it to the robot for execution.

We then compare these baselines to our KGRL algorithm outlined in Eq. (17), where an RL policy $\pi(\boldsymbol{\xi}|t)$ generates null space actions $\boldsymbol{\xi}$ that modify the trajectory using the null-space projector in LC-NS-KMP. For KGRL, the KMP input is $\boldsymbol{s} = t$ and outcome is $\boldsymbol{\eta} = \boldsymbol{p}_t$. The robot follows the resultant 2D position $\hat{\boldsymbol{p}}_t = \mathbb{E}(\boldsymbol{\eta}(t))$, derived from Eq. (17). In KGRL, safety is ensured by defining a boundary constraint on robot state as $\boldsymbol{g}_{n,1} = [-1, 0]$, $c_{n,1} = -0.6$, $\forall n \in \{1, 2, \ldots, N\}$, so that the robot does not enter into the Restricted Zone.

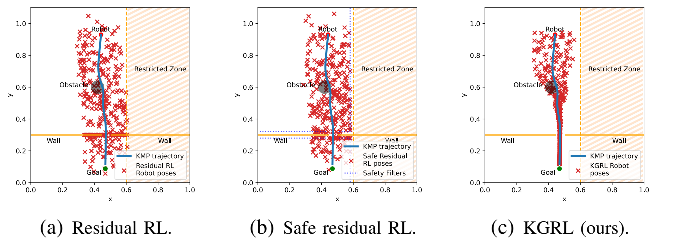
In all cases, the reward function for the robot is given by

$$r_t = r_a + r_o + r_T, \qquad r_a = -10\delta\boldsymbol{p}_t^{\top}\delta\boldsymbol{p}_t, \tag{18}$$

$$r_o = \begin{cases} -100(0.04 - \boldsymbol{d}_t), & \text{if } \boldsymbol{d}_t \leq 0.04 \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

$$r_T = \begin{cases} 200 & \text{at terminal step } T\text{, if successful} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where $\delta\boldsymbol{p}_t$ is the displacement of the robot, which can be different from the residual action $\Delta\boldsymbol{p}_t$, e.g. due to collisions with the environment, $\boldsymbol{d}_t$ is the distance of the robot from the obstacle, the terminal reward r_T is given if the episode terminates successfully, r_o is the obstacle avoidance cost, and r_a is the action cost. The episode is considered successful if the robot reaches the goal within 400 time steps. Conversely,



(a) Residual RL. (b) Safe residual RL. (c) KGRL (ours).

Fig. 4. Comparison of the robot poses resulting from isotropic exploration in residual RL approaches, (a)–(b), and uncertainty-aware exploration in KGRL, (c). For comparison purposes, manually-defined safety filters for safe residual RL keep the robot in the safe zone and prevent wall collisions in (b).

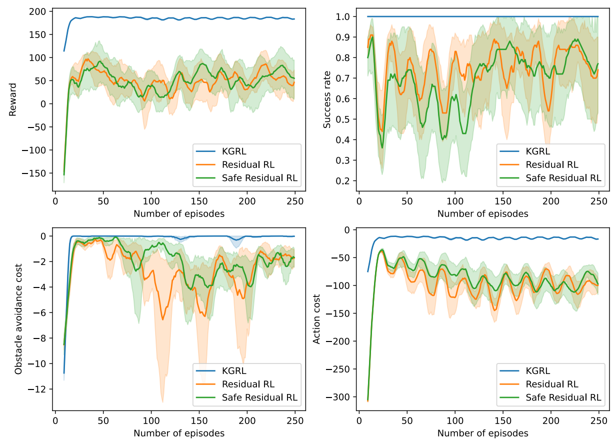


Fig. 5. Comparison of the performance of residual RL and safe residual RL with LfD to KGRL. KGRL achieves success rate of 1 from the beginning while optimizing secondary costs. Both residual RL approaches show unstable learning behavior as the robot often get stuck at the narrow passage due to inefficient exploration.

it is deemed unsuccessful if the robot becomes blocked in the narrow passage for 20 or more time steps, or if the maximum limit of 400 time steps is reached without achieving the goal.

In all cases, the RL policy is learned using a DNN with 2 hidden layers with 256 neurons each. To train the RL policies, we used the implementation of Truncated Quantile Critics (TQC) [31] from Stable-baselines3 [32], with parameters: `learning rate = 0.001`, `soft update coefficient = 0.02`, `discount factor = 0.99`, `training frequency = 8`, `gradient steps = 8`, `entropy regularization coefficient = auto`.

The performance comparison between residual RL, safe residual RL and the KGRL framework is shown in Fig. 5. KGRL quickly achieves the primary objective while minimizing both obstacle avoidance and action costs. In contrast, residual learning approaches take much longer due to isotropic noise (see Fig. 4(a) and Fig. 4(b)) used for exploration, which often causes the robot to become stuck in the narrow passage.

Conversely, KGRL modifies trajectories based on the variance in demonstrations, reducing unnecessary exploration in the low-variance region near the narrow passage (see Fig. 4(c)). Additionally, hard constraints in KGRL keep the robot within a safe zone, enhancing its overall performance. Interestingly, the walls defining the narrow passage are not modeled as hard constraints. Instead, collisions in that region are avoided through soft constraints on exploration, informed by the low variance in demonstrations where the robot remained at a safe distance from the walls.

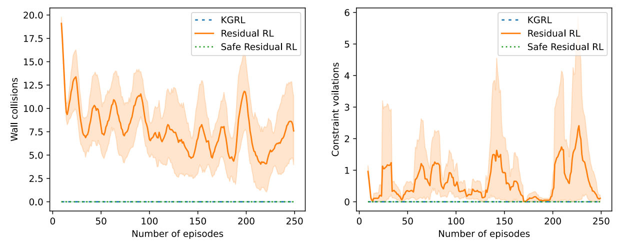


Fig. 6. Comparison of the number of wall collisions and constraint violations per episode. KGRL avoids wall collisions as the exploration is guided by low-variance in the motion near the narrow passage, while linear constraints prevent any constraint violations. Residual RL shows high number of wall collisions as well as constraint violations. Safety filters used in safe residual RL prevent wall collisions and constraint violations.

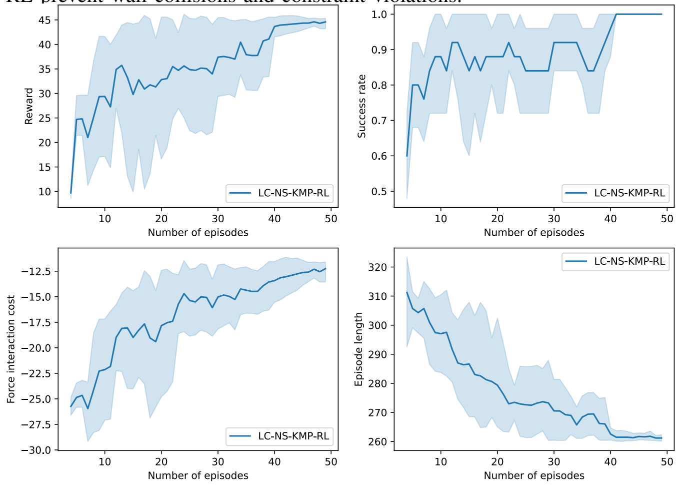


Fig. 7. Performance of KGRL on BNC connector assembly task. The robot learns to solve the task in 45 episodes on an average, simultaneously minimizing the interaction force.

Figure 6 compares the number of collisions with the wall and the constraint violations in residual RL, safe residual RL and KGRL during learning. Since demonstrations exhibit low variance while passing through the narrow passage, our approach effectively shows no collisions with the wall due to the guided exploration. As seen in Fig. 4, the magnitude of the exploration actions generated in KGRL is low in this region as it adheres to the demonstrated variance. On the other hand, the isotropic exploration noise in residual RL leads to more collisions with walls. Constraints enforced in KGRL keep the robot out of the restricted zone, effectively limiting the constraint violations to zero. Safe residual RL also shows no constraint violations and collisions with the wall due to safety filters. In this case, the safety filters are needed to be explicitly implemented near the walls, unlike KGRL where collisions with the walls are avoided due to the state-dependent uncertainty-aware exploration. With no constraint enforcement mechanism in residual RL, we observe a high number of constraint violations and wall collisions during learning. Please refer to the supplementary material for the video showing the details of the experiments.

B. Experiments on real robot

To evaluate our framework on a real robot, we selected a task from the NIST assembly benchmark 1 [5] involving the plugging of a BNC connector. This task is particularly challenging and requires multiple manipulation strategies for different phases: inserting, aligning, and locking the connector. The strategy learned from demonstration alone is insufficient to complete the task, and relying purely on RL would necessitate an impractically large number of trials. Our approach utilizes state-dependent guided exploration, allowing the robot to explore the state-action space selectively, where necessary. Additionally, linear inequality constraints reduce the state space and ensure the robot's safety. We use the DLR SARA robot in our experiment for learning the task of inserting a BNC connector. Figure 1 illustrates the experimental setup. Images (A1) and (A2) present side and top views of the BNC male connector, while (A3) and (A4) show the respective views for the BNC female connector. Images (B1) to (B4) depict the stages of picking, aligning, inserting, and locking the BNC male connector, respectively, while the human is demonstrating the task on the DLR SARA robot.

Demonstrations $D = \{\{t_{n,m}, \boldsymbol{p}_{n,m}\}_{n=1}^{400}\}_{m=1}^{5}$ were provided for the aligning, inserting, and locking phases. The 6D pose of the robot end-effector $\boldsymbol{p} = [x\ y\ z\ a_z\ a_y\ a_x]^\top$ (where a_x, a_y, a_z represent the Euler angles) is measured in the target frame, which is given by the end-effector pose when the connector is locked. For practical purposes, this target frame is assumed to be the last frame of each successful demonstration. We then learned a *vanilla* KMP [4] from D with $\boldsymbol{s} = t$ and $\boldsymbol{\eta} = \boldsymbol{p}_t$. This KMP was tested but found inadequate for task completion due to kinematic and dynamic uncertainties as well as the KMP's inability to effectively capture the contact dynamics involved in the task.

We then formulated a KGRL problem and a RL policy $\pi(\boldsymbol{\xi}|\boldsymbol{q}_t)$ was learned to complete the task, where the state for the RL policy $\boldsymbol{q}_t = [t\ \boldsymbol{p}_t^\top\ \boldsymbol{f}_t^\top]^\top$ with $\boldsymbol{f}_t$ being the 6D wrench measured at the center of compliance, $\boldsymbol{s} = t$, and $\boldsymbol{\eta} = \boldsymbol{p}$. We defined the linear inequality constraints in XY-plane so that exploration does not deviate too far from the alignment pose,

$$\begin{aligned} \boldsymbol{g}_{n,1}^\top &= [1\ 0\ 0\ 0\ 0\ 0],\ c_{n,1} = -0.002, \\ \boldsymbol{g}_{n,2}^\top &= [-1\ 0\ 0\ 0\ 0\ 0],\ c_{n,2} = -0.002, \\ \boldsymbol{g}_{n,3}^\top &= [0\ 1\ 0\ 0\ 0\ 0],\ c_{n,3} = -0.002, \\ \boldsymbol{g}_{n,4}^\top &= [0\ -1\ 0\ 0\ 0\ 0],\ c_{n,4} = -0.002,\ \forall n = 1 \dots N. \end{aligned} \tag{21}$$

Using $\boldsymbol{g}_{n,f}$ and $c_{n,f}$ formulated in eq. (21), matrices $\bar{\boldsymbol{G}}$ and $\bar{\boldsymbol{C}}$ are constructed.

The reward function for the RL agent is given by,

$$r_t = -0.01\boldsymbol{\xi}_t^\top \boldsymbol{\xi}_t - 0.01\boldsymbol{f}_t^\top \boldsymbol{f}_t + r_T, \tag{22}$$

$$r_T = \begin{cases} 60 & \text{at terminal step } T \text{ if successful,} \\ -50 & \text{if robot detects collision} \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

Since the DLR SARA robot can detect collisions observing anomalous joint torques, we use this signal in the reward function. The RL policy is learned using a DNN with 2 hidden layers with 256 neurons each and TQC for training, similarly to the simulation experiments with parameters: `learning rate = 0.001, soft update coefficient = 0.01, discount factor = 0.995, training frequency = 8, gradient steps = 8, entropy regularization coefficient = auto with initial value of 0.1`. Figure 7 illustrates the overall performance of KGRL. The robot successfully learned to insert and lock the connector in under 45 episodes while significantly reducing force interactions with the environment—an important factor for ensuring the robot's long-term safe operation. The robot achieved a success rate of 1 in 45 episodes, along with a decrease in episode length, which results in faster task completion. Please refer to the supplementary material for the video showing the details of the experiments.

V. DISCUSSION

With evaluations in simulation and on the real robot, we demonstrated that a complex task can be learned using KGRL even with a sparse reward function in a sample-efficient and safe manner. With KGRL, a simulated robot learns to achieve the primary goal of reaching the target while minimizing secondary costs, showing success rate of 1 from the beginning. Meanwhile, both residual RL approaches show unstable learning behavior with the same reward function, as shown Fig. 5. In both residual RL approaches, isotropic exploration noise leads to counter-productive exploration near the narrow passage. Also, despite improving safety compared to the π_{res} baseline, the definition of π_{safe} comes at the cost of having to manually define wall constraints for the task to succeed, which is not required in KGRL, since the exploration behavior is extracted from the data.

As discussed above, KGRL is capable of learning complex contact tasks directly on the real robot. The safety of both robot and environment is facilitated by the hard constraints defined in the framework. For the task of inserting a BNC connector, a policy learned from demonstrations using KMP alone was not sufficient. With KGRL, we showed that the demonstrations can be used for accelerating RL by extracting the task completion strategy along with the exploration strategy. With the possibility of learning the task on a real robot safely and in a time-efficient manner, we largely alleviate the need of modeling the task meticulously in simulation. Furthermore, we use an off-the-shelf off-policy algorithm for learning the tasks. Meticulous hyper-parameter tuning was not necessary in our approach for learning the tasks successfully.

In Section III-C, we discussed the properties of LC-NS-KMP. Given its ability to adapt based on demonstration variance while respecting constraints, LC-NS-KMP is not limited to RL and also offers desirable properties for LfD, especially on real robots. Also note that we prioritized success rate and learning efficiency in our evaluations, and therefore did not explicitly focus on smooth exploration. Nevertheless, the framework accommodates additional velocity constraints–such as those proposed in [30] in the context of LfD–which can be defined to promote smoother exploratory behavior.

We would also like to highlight some limitations of our framework. The safety constraints in this work are manually defined and require expert knowledge during design; however, such constraints could also be extracted from demonstration data, e.g., upper and lower limits of motion in each degree of freedom. Another possible limitation is that we assume that the demonstrations encode an appropriate exploration strategy, which might not hold true in some cases. In such scenarios, a potential mitigation strategy is to manually tune the covariance matrix $\boldsymbol{\Sigma}$, as it is human-interpretable, leveraging the flexibility inherent in our approach.

VI. Conclusion

In this paper, we presented a novel movement primitive representation called Linealy Constrained Null-space Kernelized Movement Primitives (LC-NS-KMP), which can learn movement primitives from demonstrations allowing modifications through null-space actions, while respecting the linear inequality constraints. We leverage this movement primitive representation to deliver a novel constrained and guided RL method called Kernelized Guided Reinforcement Learning (KGRL). We evaluated our approach to highlight the effectiveness of KGRL in learning challenging manipulation tasks involving complex contacts directly on real robot. By integrating state-dependent guided exploration and linear inequality constraints, we were able to facilitate efficient learning and enhance the robot's operational safety. Our approach enables the robot to master connector insertion and locking in under 45 episodes, significantly reducing learning time and effort. Additionally, the reduction in force interactions with the environment indicates a pathway toward long-term reliability and safety in robot manipulation. Future work includes automating constraint extraction and incorporating non-linear constraints to extend applicability to broader assembly challenges.

Acknowledgments

We would like to thank Antonin Raffin for the discussions on the off-policy RL algorithms and assisting with Stable Baselines 3 framework.

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[2] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[3] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

[4] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *International Journal of Robotics Research (IJRR)*, vol. 38, no. 7, pp. 833–852, 2019.

[5] K. Kimble, K. Van Wyk, J. Falco, E. Messina, Y. Sun, M. Shibata, W. Uemura, and Y. Yokokohji, "Benchmarking protocols for evaluating small parts robotic assembly systems," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 883–889, 2020.

[6] M. Iskandar, C. Ott, O. Eiberger, M. Keppler, A. Albu-Schäffer, and A. Dietrich, "Joint-level control of the dlr lightweight robot sara," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 8903–8910.

[7] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153 171–153 187, 2021.

[8] A. Padalkar, G. Quere, A. Raffin, J. Silvério, and F. Stulp, "Guiding real-world reinforcement learning for in-contact manipulation tasks with shared control templates," *Autonomous Robots*, vol. 48, no. 4, p. 12, 2024.

[9] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," *arXiv preprint arXiv:2107.10253*, 2021.

[10] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[11] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[12] J. Eßer, N. Bach, C. Jestel, O. Urbann, and S. Kerner, "Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics [survey]," *IEEE Robotics & Automation Magazine (RAM)*, vol. 30, no. 2, pp. 67–85, 2023.

[13] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[14] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 6292–6299.

[15] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[16] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[17] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[18] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid, "Residual reinforcement learning from demonstrations," *arXiv preprint arXiv:2106.08050*, 2021.

[19] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, "Residual learning from demonstration: Adapting dmps for contact-rich manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4488–4495, 2022.

[20] J.-P. Sleiman, M. Mittal, and M. Hutter, "Guided reinforcement learning for robust multi-contact loco-manipulation," in *8th Annual Conference on Robot Learning (CoRL 2024)*, 2024.

[21] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[22] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6236–6243.

[23] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[24] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International conference on machine learning (ICML)*, 2015, pp. 997–1005.

[25] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," *Advances in neural information processing systems*, vol. 29, 2016.

[26] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research (IJRR)*, 2024.

[27] Z. Zhu, H. Zhao, H. He, Y. Zhong, S. Zhang, H. Guo, T. Chen, and W. Zhang, "Diffusion models for reinforcement learning: A survey," *arXiv preprint arXiv:2311.01223*, 2023.

[28] Y. Zheng, J. Li, D. Yu, Y. Yang, S. E. Li, X. Zhan, and J. Liu, "Safe offline reinforcement learning with feasibility-guided diffusion model," in *The Twelfth International Conference on Learning Representations*.

[29] J. Silvério and Y. Huang, "A non-parametric skill representation with soft null space projectors for fast generalization," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2023, pp. 2988–2994.

[30] Y. Huang and D. G. Caldwell, "A linearly constrained nonparametric framework for imitation learning," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2020, pp. 4400–4406.

[31] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *International Conference on Machine Learning (ICML)*, 2020, pp. 5556–5566.

[32] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.