IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

MoRe-ERL: Learning Motion Residuals using Episodic Reinforcement Learning

Xi Huang, Hongyi Zhou, Ge Li, Yucheng Tang, Weiran Liao, Björn Hein, Tamim Asfour and Rudolf Lioutikov

***Abstract*—We propose MoRe-ERL, a framework that combines Episodic Reinforcement Learning (ERL) and residual learning, which refines preplanned reference trajectories into safe, feasible, and efficient task-specific trajectories. This framework is general enough to incorporate into arbitrary ERL methods and motion generators seamlessly. MoRe-ERL identifies trajectory segments requiring modification while preserving critical task-related maneuvers. Then it generates smooth residual adjustments using B-Spline-based movement primitives to ensure adaptability to dynamic task contexts and smoothness in trajectory refinement. Experimental results demonstrate that residual learning significantly outperforms training from scratch using ERL methods, achieving superior sample efficiency and task performance. Hardware evaluations further validate the framework, showing that policies trained in simulation can be directly deployed in real-world systems, exhibiting a minimal sim-to-real gap.**

***Index Terms*—Motion and Path Planning; Reinforcement Learning; Integrated Planning and Learning**

I. Introduction

ROBOTIC applications, such as multi-arm cooperation, often require frequent motion adaptation to ensure safety and task efficiency. The adaptation must be reactive, smooth, and feasible, with algorithms responding rapidly. Moreover, it is desirable that the new trajectories consider system dynamics, such as potential environmental changes. Taking these changes into account reduces the need for repeated online adaptations. Episode-based Reinforcement Learning (ERL) methods [1]–[4] have shown success in addressing complex tasks. Based on the initial observations, the ERL methods generate full trajectories in deterministic computation time by parameterizing the movement primitives (MPs), accounting for the possible changes and interactions in the environment, ensuring smooth execution across an episode. Recent developments in MPs [5], [6] support various boundary conditions, enabling smooth transitions during trajectory switches. Replanning techniques [3] further allow trajectory updates as the episode progresses. This paper proposes MoRe-ERL (**Mo**tion **Re**siduals using **ERL**), a general framework that generates motion residuals to refine previously planned task-related reference trajectories. Given a reference trajectory, MoRe-ERL identifies the trajectory segments that require modification while preserving critical task-related behaviors. Trajectory refinements for these segments are generated using B-Spline-based movement primitives to ensure smooth transitions. Three refinement strategies are investigated, and their performance is evaluated in multiple simulation tasks. The segment identification and the trajectory refinements are jointly learned as a correlated policy. Having the reference trajectory as prior knowledge, MoRe-ERL significantly outperforms training from scratch using ERL methods, achieving superior sample efficiency and task performance. Our main contributions are

- To the best of our knowledge, the first RL algorithm that combines ERL and residual learning, achieving superior sample efficiency and task performance.
- An end-to-end policy that identifies the segments of reference trajectories needing modification and parameterizes movement primitives as residuals.
- Three novel trajectory refinement strategies using B-spline-based movement primitives, which enforce smooth transition between the reference and the fixed trajectories.

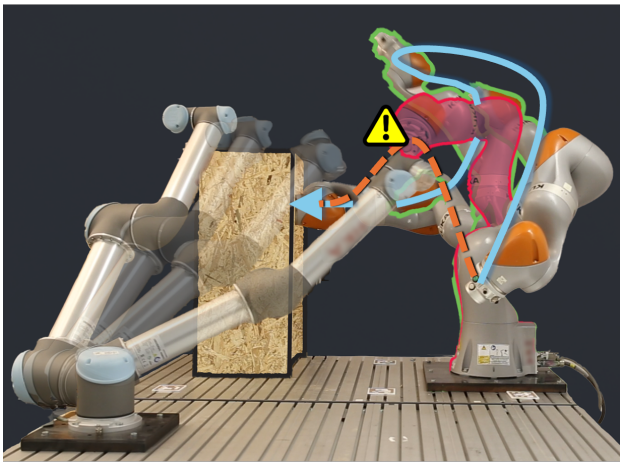


Fig. 1: MoRe-ERL applies residuals to adjust the trajectory of a KUKA iiwa robot into a safe and feasible motion, effectively avoiding the moving UR5 robot. Snapshots of the KUKA iiwa executing the adjusted motion are outlined in green, while the red marker highlights the frame on the reference trajectory where the robots would have collided.

II. Related Works

A. Episodic Reinforcement Learning

ERL is a distinct family of RL that emphasizes the maximization of returns over entire episodes, typically lasting several seconds, rather than optimizing each step in the episodes while interacting with the environment [7]. Unlike Step-based RL, such as PPO [8] and SAC [9], ERL shifts the solution

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany {x.huang, hongyi.zhou, ge.li, weiran.liao, bjoern.hein, asfour, lioutikov}@kit.edu, yucheng.tang@partner.kit.edu

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

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

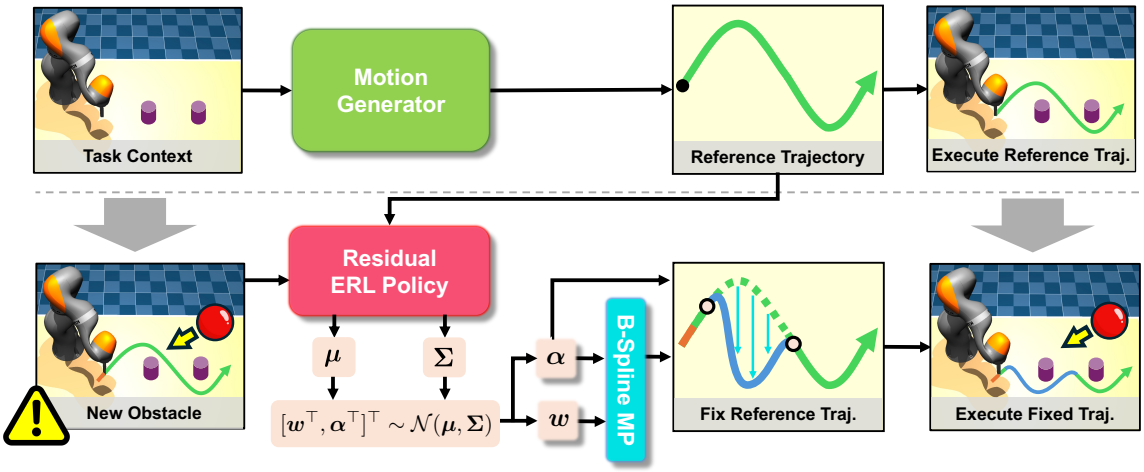


Fig. 2: Illustration of the MoRe-ERL pipeline. The robot follows the reference trajectory provided by a motion generator, with the executed segment shown in — and the remainder in —, based on the task context. When the task context changes, such as the appearance of new obstacles, MoRe-ERL identifies critical segments on the remaining reference trajectory (—) using learned parameters $\boldsymbol{\alpha} = [\alpha_s, \alpha_e]^\top$ and parameterize residuals $f(\boldsymbol{w})$ for the selected segments using B-spline-based movement primitives. The adjusted trajectory, after applying these residuals, is shown by the solid blue-green curve (—).

search from per-step actions to a parameterized trajectory space, leveraging techniques like MPs [10], [11] for generating action sequences. This approach enables a broader exploration horizon [2]. Recent advances in *Deep Black-Box Reinforcement Learning* (BBRL) [4] have integrated ERL with deep neural networks. MoRe-ERL employs ERL to learn motion refinements and improve the sample efficiency by leveraging prior knowledge from pre-planned reference trajectories.

B. Learning Motion Residual with RL

The idea of using RL to learn motion residual [12], [13] leverages the generalization capability of RL to handle complex dynamics like contacts and friction. At the same time, learning motion residuals simplifies the RL problem, thereby reducing the demand for samples and making real-world applications more feasible [13], [14]. Recent works have also explored incorporating MPs into residual RL [15], [16], where MPs serve as a base motion generator, and step-based RLs are used to learn the motion residual for every control step. For example, model-free RL generates step-based residuals for a DMP-based policy using only task observations [16]. Similarly, ProMP-based policies serve as fixed nominal policies, with step-based agents computing residuals in a low-frequency control loop [16]. However, step-based residual RL suffers from similar limitations to step-based RL, including a lack of smoothness in generated motions and a heavy reliance on dense Markovian reward. In contrast, MoRe-ERL takes reference trajectories from arbitrary base policies as input and uses BMPs to parameterize residual sequences. It fully leverages BMPs by allowing partial refinement of the reference trajectory rather than modifying it entirely, thereby preserving critical task behaviors. Consequently, the length of the residual sequence is flexible and determined by the policy. Compared to DMPs [10] and ProMPs [11], BMPs mathematically guarantee enforcement of initial and terminal conditions, making them well-suited for generating motion residuals. To our knowledge, MoRe-ERL is the first framework to combine BMPs with ERL for parameterizing *residual sequences*.

C. Motion Planning in Robotics

Two important categories in motion planning methods are sampling-based and optimization-based motion planning. In a dynamic environment, sampling-based methods usually address problems using reactive methods or fully taking into account the states of the environment in the future. Reactive methods [17]–[19] enable fast replanning but neglect temporal aspects, often producing solutions that soon become invalid. On the other hand, methods such as ST-RRT* [20] extend the spatial space and take into account the temporal aspect. They assume full knowledge of all paths of participants in the environment. In real-world scenarios, however, full knowledge of the environment in the future is usually not accessible. Optimization-based methods [21], [22] directly output a trajectory with parameterized velocity and acceleration profile regarding user-defined constraints. Methods of this category can naturally support the temporal aspect when providing full knowledge of how the environment evolves. In MoRe-ERL, motion generators serve the role of base planner and provide reference trajectories in an offline manner. MoRe-ERL is base-planner-agnostic and refines trajectories during execution.

III. PRELIMINARIES

A. Episodic Reinforcement Learning (ERL)

ERL [2], [23] predicts an entire sequence of actions to accomplish a task by optimizing cumulative rewards without explicitly modeling detailed state transitions within an episode. ERL methods usually predict a weight vector $\boldsymbol{w}$ given the task context. This vector is then used to parameterize a complete trajectory $\boldsymbol{y}(t) = f(\boldsymbol{w})$ for $\boldsymbol{y}(t) \in \mathbb{R}^D$ and $t \in [0, T]$, where D corresponds to the dimensionality of the trajectory

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

space, such as the DoFs in a robotic system, T represents the trajectory duration, and $f[\cdot]$ indicates a generic function for trajectory parameterization using a motion generator. The predicted trajectory can either be directly utilized as per-step actions or serve as input to a trajectory tracking controller for computing lower-level motor commands. Given the initial state $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$ specifying the starting configuration and task context, the goal of ERL is to find a weight vector $\boldsymbol{w}$ that maximizes the return $R(\boldsymbol{s}_0, f(\boldsymbol{w}))$ after executing the trajectory $\boldsymbol{y}(t) = f(\boldsymbol{w})$. The ERL learning objective is generally expressed as:

$$J = \mathbb{E}_{p(\boldsymbol{s}_0),\pi_\theta(\boldsymbol{w}|\boldsymbol{s}_0)}\left[R(\boldsymbol{s}_0, f(\boldsymbol{w})) - V_\phi(\boldsymbol{s}_0)\right], \tag{1}$$

where π_θ denotes the policy parameterized by θ, often implemented using a neural network. The return $R(\boldsymbol{s}_0, f(\boldsymbol{w})) = \sum_{t=0}^{T} \gamma^t r_t$ is the cumulative reward obtained by following the trajectory, where γ is the discount factor, and r_t is the reward at time step t. The term $V_\phi(\boldsymbol{s}_0)$ represents a value estimator of the state $\boldsymbol{s}_0$, parameterized by ϕ, and acts as a baseline to stabilize training [24].

When compared to traditional step-based RL (SRL) methods like PPO [8], ERL shifts the solution search from the per-step action space $\mathcal{A}$ to a parameterized trajectory space $\mathcal{W}$, predicting trajectory parameters as $\pi(\boldsymbol{w}|\boldsymbol{s})$. This often facilitates broader exploration and results in smooth, correlated motion trajectories. Additionally, the learning objective in Eq. (1) relaxes the requirement for Markovian rewards [24], which enforces that the reward r_t at a given time step t depends only on the current state $\boldsymbol{s}$ and action $\boldsymbol{a}$. Step-based RL methods such as SAC [9] rely on temporal difference (TD) learning. This requires Markovian rewards in order to assign value credits to per-step actions and states properly. In contrast, ERL assigns task credit to the entire trajectory episode parameterized by $\boldsymbol{w}$ by aggregating per-step rewards. This removes the requirement for rewards to be Markovian, allowing for the use of delayed or history-dependent rewards, referred to as *non-Markovian rewards* [24]. Intuitively, non-Markovian rewards offer greater flexibility and simplicity in task design [4], as they rely on fewer assumptions compared to their Markovian counterparts.

Usually, ERL methods use MPs as the motion generator. MPs can encapsulate trajectories from a lower-dimensional parameter space, thereby reducing the problem complexity.

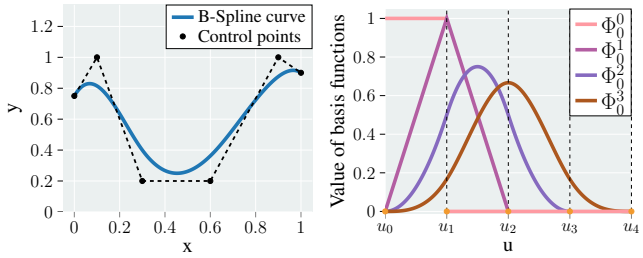
B. Using Movement Primitives in ERL

Parameterizing trajectories using MPs [5], [10], [11] is central to ERL methods. We first describe Probabilistic Movement Primitives (ProMPs) and then introduce BMPs using the same formalism as ProMPs.

Probabilistic Movement Primitives (ProMPs) [11] represent a trajectory $\boldsymbol{y}(t)$ using a linear basis function:

$$\boldsymbol{y}(t) = f(\boldsymbol{w}) = \boldsymbol{\Phi}(\frac{t}{T})^\top \boldsymbol{\omega} = \boldsymbol{\Phi}(u)^\top \boldsymbol{\omega}, \tag{2}$$

where $u = t/T \in [0, 1]$ denotes the normalized time, also called the phase variable. The term $\boldsymbol{\Phi}(u) = [\Phi_1(u), \Phi_2(u), ..., \Phi_N(u)]^\top$ represents N basis functions



(a) B-Spline in 2D (b) Basis functions

Fig. 3: Illustration of BMPs: (a) A clamped B-spline curve in 2D parameterized with 6 control points. (b) Basis function of different orders using recursive formulation, where Φ_0^p denotes the basis function of p^{th} order for the 0^{th} control point. The knots u represent the change of time.

for each DoF, evaluated at u. The weight vector $\boldsymbol{w} = [w_1, w_2, ..., w_N]^\top$ controls the trajectory shape by scaling the basis functions. Typically, a neural network is used to predict the mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ and the weight vector is sampled from the distribution $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})$. For non-periodic trajectories, ProMPs often utilize radial basis functions (RBF) as the basis functions, with their centers uniformly distributed in the phase space $[0, 1]$.

ProMPs are advantageous due to their simple linear representation, enabling fast computation and probabilistic modeling [11]. However, ProMPs lack mathematical support for enforcing specific boundary conditions at the trajectory's start and end points. This limitation restricts their ability to generate new trajectories that seamlessly transition from an existing one. However, this is a critical requirement for real-world scenarios, where frequent trajectory switching is necessary. Recent works [6], [25] address these inherent limitations by replacing the RBF functions in ProMPs with B-splines. The resulting model B-spline-based Movement Primitives (BMP) retains the linear basis function representation of ProMPs while supporting an arbitrary number and order of trajectory transition conditions. Mathematically, these conditions are known as boundary conditions.

Definition of BMP. The basis functions of BMP, $\boldsymbol{\Phi}^P(u) = [\Phi_1^P(u), \Phi_2^P(u), ..., \Phi_N^P(u)]^\top$, are defined as P-th order polynomial functions, where $0 \leq P < N$. These basis functions are constructed over M definition intervals, equally divided by $M+1$ knots $u_0, ..., u_M$, with $u_0 = 0$ and $u_M = 1$. Typically, $M = N + P$ [26] and the intervals between two adjacent knots have the same length δ. In the context of B-splines, the weights $\boldsymbol{w}$ are also interpreted as *control points*, which define a convex hull that bounds the trajectory, see Fig. 3a.

Each basis function Φ_n^P, where $n \in [1, N]$, is defined recursively from order 0 to order P [26]. To illustrate this recursive process, we denote intermediate orders with the index p, where $p \in [0, P]$. For $p = 0$, the basis functions are piecewise constant:

$$\Phi_n^{p=0}(u) = \begin{cases} 1 & \text{if } u_n \leq u < u_{n+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

For $p > 0$, each basis function $\Phi_n^p(u)$ is computed by

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Type	Condition	$w_1^{(i)}$ = ?
Position	$w_1^{(0)} = y(t_0)$	$w_1^{(0)}$
Velocity	$w_1^{(1)} = \dot{y}(t_0)$	$\frac{P}{\delta}(w_2^{(0)} - w_1^{(0)})$
Acceleration	$w_1^{(2)} = \ddot{y}(t_0)$	$\frac{P(P-1)}{\delta^2}(w_3^{(0)} - 2w_2^{(0)} + w_1^{(0)})$

TABLE I: Mapping between boundary conditions and control points at t_0. We present the control point $w_1^{(i)}, i = 0, 1, 2$ given initial position $y(t_0)$, velocity $\dot{y}(t_0)$ and acceleration $\ddot{y}(t_0)$ conditions, respectively. By recursively applying Eq. (6) and (7), the higher-order control point $w_1^{(i)}$ can be eventually represented using the 0-th order control points.

interpolating between two corresponding lower-order basis functions, $\Phi_n^{p-1}(u)$ and $\Phi_{n+1}^{p-1}(u)$, using coefficients j_n and j_{n+1}:

$$\Phi_n^p(u) = j_n\ \Phi_n^{p-1}(u)\ +\ j_{n+1}\ \Phi_{n+1}^{p-1}(u). \tag{4}$$

$$j_n = \frac{u - u_n}{p\ \delta},\quad j_{n+1} = \frac{u_{n+p+1} - u}{p\ \delta} \tag{5}$$

By recursively applying Eq. (4) until order P, the P-th order basis function is obtained, see Fig. 3b. By substituting the resulting BMP basis functions into Eq. (2), the trajectory of BMP can be computed using the linear basis function representation of ProMPs.

Derivative of B-Splines. It is worth noting that the i-th order derivative of a P-th order B-spline remains a $(P-i)$-th order B-spline:

$$\boldsymbol{y}^{(i)}(t) = \boldsymbol{\Phi}^{P-i}(u)^\top \boldsymbol{\omega}^{(i)}, \tag{6}$$

where $\boldsymbol{\omega}^{(i)} = [w_1^{(i)}, ..., w_{N-i}^{(i)}]^\top$ represents the control points of the B-spline's derivatives. These control points are computed recursively:

$$w_n^{(i)} = \frac{P-i}{\delta}[w_{n+1}^{(i-1)} - w_n^{(i-1)}]. \tag{7}$$

Enforcing boundary conditions of arbitrary orders. To ensure the trajectory passes through given starting and ending positions, BMP employs clamped B-splines [26] where the trajectory goes through the first and the last control points. Thus, we directly align these two control points with the given position values. Similarly, control points derived from Eq. (6) and (7) enforce higher-order conditions, such as velocity and acceleration. In Table I, we summarize the mapping between boundary conditions at t_0 and the corresponding control points. Ending conditions follow similar mappings.

IV. Method

A. Problem Definition

For an task context $\boldsymbol{c}_0$, a robot trajectory $\boldsymbol{y}_{r,0:T} = \{\boldsymbol{y}_{r,0}, \ldots, \boldsymbol{y}_{r,T}\}$ is generated solve this task. As the task context changed to $\boldsymbol{c}_\tau$ at time $\tau \in (0, T]$, this work aims to find a policy that generates trajectory-level refinements, $\Delta\boldsymbol{y}_{\tau:T}$, based on the current context $\boldsymbol{c}_\tau$ and original robot trajectory $\boldsymbol{y}_{r,\tau:T}$. A state encoder $s(\cdot)$ is used to encode $\boldsymbol{c}_\tau$ and $\boldsymbol{y}_{r,\tau:T}$ into a state vector $\boldsymbol{s}_\tau = s(\boldsymbol{c}_\tau, \boldsymbol{y}_{r,\tau:T})$. Following the previous section III-A, an ERL problem is formulated to optimize the policy distribution $\pi(\boldsymbol{w}|\boldsymbol{s}_\tau)$ by maximizing the expected roll-out return $R(\boldsymbol{s}_\tau, \boldsymbol{w})$ of an episode using the following objective function

$$J = \mathbb{E}_{p(\boldsymbol{s}_\tau), \pi_\theta(\boldsymbol{w}|\boldsymbol{s}_\tau)}\ \left[R(\boldsymbol{s}_\tau, \boldsymbol{w}) - V_\phi(\boldsymbol{s}_\tau)\right], \tag{8}$$

where the vector $\boldsymbol{w}$ determines the start and the end of the residual action as well as the parameterization of the action sequence using BMPs. While MoRe-ERL adopts BBRL [4] for the optimization, the framework is modular enough to plug in any other ERL algorithms.

B. Learning Residuals for Reference Trajectories

Reference trajectories contain information for completing the given tasks and can be accessed from various sources, such as sampling-based motion planning or other learned policies. This information serves as a valuable prior in two key aspects: (a) it reduces the complexity of learning, thereby improving sample efficiency, and (b) it preserves dexterous robot behaviors that are challenging to learn from scratch.

When the task context changes to $\boldsymbol{c}_\tau$ at time τ unexpectedly, e.g., the environment didn't evolve as anticipated, the policy generates trajectory-level refinements $\Delta\boldsymbol{y}_{\tau:T}$ based on the context $\boldsymbol{c}_\tau$ and the reference trajectory $\boldsymbol{y}_{r,\tau:T}$, see Fig. 2. The reference trajectory $\boldsymbol{y}_{r,\tau:T}$ is a partially executed trajectory grounded on the previous context $\boldsymbol{c}_t$ at time $t < \tau$. The refinements are applied on top of the reference trajectory,

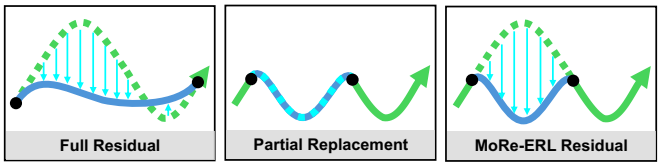
$$\boldsymbol{y}_{\tau:T} = \boldsymbol{y}_{r,\tau:T} + \Delta\boldsymbol{y}_{\tau:T}. \tag{9}$$

To allow for greater flexibility in modifying the reference trajectory, we introduce two additional timing variables, $\alpha_s, \alpha_e \in [\tau, T]$ with $\alpha_s \leq \alpha_e$, marking the refinement's start and end on the reference trajectory. Using these variables, the refinements at time $k \in [\tau, T]$ are defined as:

$$\Delta\boldsymbol{y}_{\tau:T} = \begin{cases} \Delta\boldsymbol{y}_k & \text{for } k \in [\alpha_s, \alpha_e] \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Fig. 4 (right) visualizes the refinements described in Eq. (10), termed as *MoRe-ERL residuals*. The reference trajectory is partially modified by the learned residuals $\Delta\boldsymbol{y}_k$ between α_s and α_e. These two timing variables are represented by bold solid points. To ensure continuity and smoothness at α_s and α_e during trajectory switching, we use BMPs to parameterize the residual. BMPs enforce boundary conditions up to arbitrary orders. In this case, the boundary conditions are set to be $\mathbf{0}$. This ensures that the position and velocity of the refined trajectory align with the reference trajectory α_s and α_e, guaranteeing the continuity and smoothness on trajectory switches. To be more specific, to parameterize a residual trajectory $\Delta\boldsymbol{y}_{\alpha_s:\alpha_e}$ using N control points $\boldsymbol{w}_{1:N} = [w_1, ..., w_N]^T$, BMPs use w_1 and w_2 to enforce the boundary conditions at the start of the refinement, and use w_{N-1} and w_N at the end. The remaining control points $\boldsymbol{w}_{3:N-2} = [w_3, ..., w_{N-2}]^T$ parametrize the transition behavior $\Delta\boldsymbol{y}_{\alpha_s:\alpha_e}$. The control points $\boldsymbol{w}_{3:N-2}$ are jointly learned with α_s and α_e. Given the encoded state $\boldsymbol{s}_\tau$, the policy $\pi(\boldsymbol{w}, \boldsymbol{\alpha}|\boldsymbol{s}_\tau)$ returns the mean $\boldsymbol{\mu}_{\boldsymbol{w},\boldsymbol{\alpha}}$ and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{w},\boldsymbol{\alpha}}$ of a single Gaussian distribution. Sampling from this

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



Type	Learned Parameters	How to Refine
Full Residual	$\boldsymbol{w}$	Residuals on top of the reference
Partial Replacement	$\alpha_s, \alpha_e, \boldsymbol{w}$	Replace the reference
MoRe-ERL Residual	$\alpha_s, \alpha_e, \boldsymbol{w}$	Residuals on top of the reference

Fig. 4: MoRe-ERL residuals and two ablation variants. The reference trajectory is shown in green, with bold solid points indicating the timing variables α_s and α_e. Cyan sections show learned residuals or replacements, and the solid blue-green curve denotes the adjusted trajectory.

Gaussian distribution $[\boldsymbol{w}_{3:N-2}^\top, \alpha_s, \alpha_e]^\top \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{w},\boldsymbol{\alpha}}, \boldsymbol{\Sigma}_{\boldsymbol{w},\boldsymbol{\alpha}})$, the residual sequence can be expressed as

$$\Delta \boldsymbol{y}_{\alpha_s:\alpha_e} = \boldsymbol{\Phi}_{\alpha_s:\alpha_e}^\top \boldsymbol{w}_{3:N-2}. \tag{11}$$

Fig. 5 illustrates how the MoRe-ERL residual is applied to the reference trajectory. Different from the jerky motions produced by step-based methods, BMPs ensure smooth transitions between the reference trajectory and the refinements.

Refinement Variants. In addition to partially applying residuals, we consider two alternative approaches, termed *full residual* and *partial replacement*. The full residual variant is a special case of Eq. (10) with $\alpha_s = \tau$ and $\alpha_e = T$, applying residuals to the entire trajectory (Fig. 4, left). The partial replacement variant partially replaces segments between α_s and α_e with the refinement sequences (see Fig. 4, middle). The partial replacement modifies the reference trajectory as:

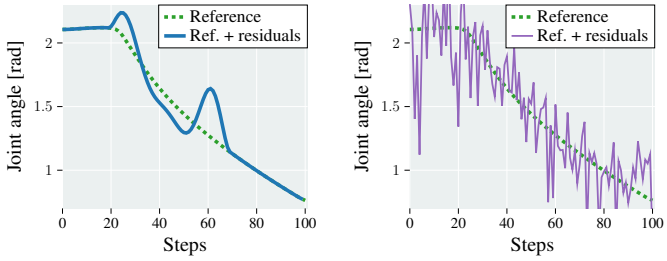
$$\boldsymbol{y}_{\tau:T} = \begin{cases} \Delta \boldsymbol{y}_k & \text{for } k \in [\alpha_s, \alpha_e] \\ \boldsymbol{y}_{r,k} & \text{otherwise.} \end{cases} \tag{12}$$

When parameterizing trajectories with BMPs in partial replacement, boundary conditions are set to match the position and velocity of the reference trajectory at α_s and α_e. We exclude full replacement from consideration as it equates to learning the trajectory from scratch.

Among MoRe-ERL residuals and these two variants, MoRe-ERL residuals demonstrate the best overall performance across various scenarios. Learning residuals leverages prior knowledge embedded in reference trajectories, preserving critical maneuvers and enhancing task completion. The identification of α_s and α_e contributes to retaining essential behaviors. Further details are provided in Section V.

V. Experiments

More-ERL is evaluated in two simulation experiments in MuJoCo [27] and an experiment with real-world hardware. In both simulation scenarios, baseline methods include sampling-based motion planning methods, episode-based RL methods,



(a) MoRe-ERL residuals (b) Step-based residuals

Fig. 5: Random roll-out using MoRe-ERL and step-based residual method. In the demonstrated case, the trajectory with MoRe-ERL residuals (–) deviates from the reference trajectory at $\alpha_s = 20$ and converges back at $\alpha_e = 70$.

and step-based RL methods. While the episode-based methods work perfectly with non-Markovian rewards, the step-based methods perform poorly in such settings [4]. For an *unfavourable* comparison against our method, we shape a Markovian reward grounded on the performance of step-based methods and evaluate MoRe-ERL on both Markovian and non-Markovian rewards. Despite this *unfavourable* setting, MoRe-ERL shows comparable performance with Markovian rewards in both tasks and achieves superior performance with non-Markovian rewards.

The Markovian returns of an episode with n_e steps summarize rewards from each step with a discount factor γ

$$R_M = \sum_{t=\tau}^{n_e} \gamma^t (\beta_c r_{c,t} + \beta_g r_{g,t} + \beta_l r_{l,t}), \tag{13}$$

where $r_{c,t}$, $r_{g,t}$, and $r_{l,t}$ indicate the reward terms at t regarding collision, goal reaching, and joint limit violation, respectively. The collision reward is assigned $r_{c,t} = -1$ when a collision occurs. The other two terms are both computed using L2-Norm. These reward terms are weighted by corresponding coefficients $\beta_{[\cdot]}$. The non-Markovian return R_{NM} does not collect rewards regarding collision and goal reaching at every step, but at the end of the episode

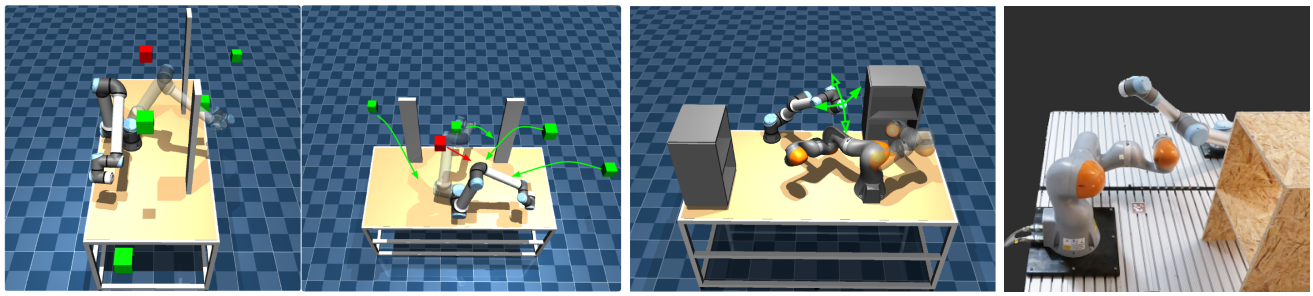
$$R_{NM} = \underbrace{\beta_c r_c + \beta_g r_g}_{\text{Non-Markovian}} + \beta_l \sum_{i}^{n_e} \gamma^i r_{l,t}. \tag{14}$$

This setting links the reward closer to the definition of success and avoids potential reward hacking. The results of both simulation scenarios show that MoRe-ERL with non-Markovian rewards achieved a significantly higher performance than baselines, see Fig. 7 and TABLE II. A video for simulation and real-world experiments is attached to the multimedia materials. Parameters for RL and BMPs are selected based on a grid search over the parameter space.

A. Multi-Box

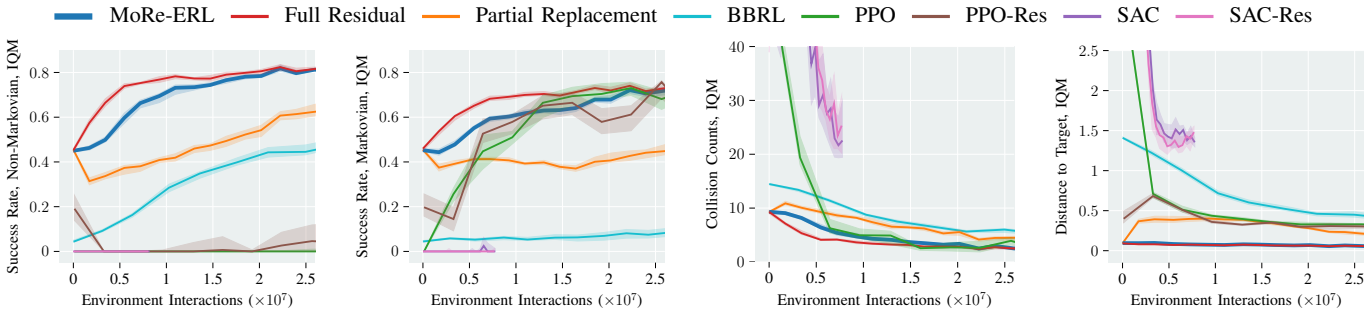
The multi-box scenario has an UR10e robot mounted on a table, which travels among three regions separated by two bars to complete arbitrary tasks, see Fig. 6a. While the robot is operating, dynamic obstacles enter the robot's working space, either moving with constant velocity or following parabolic

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

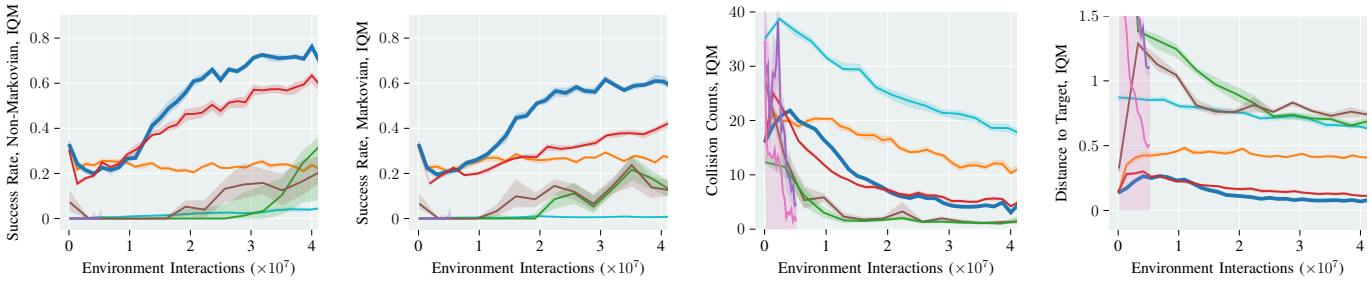


(a) Multi-box (b) Dual-arm (c) Real-world hardware setup

Fig. 6: Simulation experiments in Mujoco and real-world experiment setup. The robot with reduced opacity in simulation indicates the desired goal. In (a), green boxes follow parabolic trajectories, while the red box moves at a constant velocity. Each box is released at a different timestamp. In (b), the UR5 robot moves in the directions shown by the green arrows.



(a) Multi-box experiment



(b) Dual-arm experiment

Fig. 7: From left to right: (1) Success rate with non-Markovian reward; (2) Success rate with Markovian reward; (3) Collision counts; (4) Distance to target. Better results from the two reward settings are shown in (3) and (4). MoRe-ERL and its variant full residual show an advantage with respect to sample efficiency and task performance compared to baseline methods in both tasks, even with Markovian rewards. In the multi-box experiment, the full residual variant achieves better sample efficiency due to its smaller search space. In the scenarios requiring more dexterous behaviors, such as the dual-arm task, MoRe-ERL's ability to identify the residual interval shows clear advantages compared to applying residual to the entire trajectory.

paths. In every episode, the initial joint configurations and goal are randomly selected. The obstacles are released to move at different time stamps. The trajectories of the obstacles are designed to be *adversarial* to our method that at least one of the obstacles will hit the robot if the robot follows the reference trajectory. The episode terminates when the goal or the pre-defined maximum duration of 3 seconds is reached. For episode-based methods, the observation includes (a) the current robot configurations and velocity, (b) the goal of the robot, and (c) the parameters from which the agent can infer the trajectories of the dynamic obstacles, such as their position, velocity, and end position. The end positions of the obstacles serve the purpose of distinguishing the parabolic trajectories from the ones with constant velocity. On the other hand, the step-based methods receive a new observation at every simulation step and return the next joint position as an action. The observation for step-based methods additionally includes the current timestamp.

The residual version of both episode-based and step-based methods must be aware of the reference, on which the residual acts. Reference trajectories is generated by RRT-Connect [28]. A representation of the reference is included in the observation of the residual methods. The ERL residual methods use five intermediate waypoints of the reference trajectory as the representation, while the step-based methods use the next reference action. The reference trajectories are generated using

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

a sampling-based motion planner. Note that this approach is agnostic to the choice of planner, allowing an alternative motion generator to be seamlessly integrated.

The results of baseline methods and ablations are summarized in TABLE II, collected with a single core on an Intel i9-9900K CPU. Sampling-based methods were allocated 1 second of planning time for each problem in a space-time state space ($\mathbb{R}^{6+1}$), with 6 DoFs for robot joints and 1 DoF for time. For ST-RRT*, the maximum arrival time was set to 3 seconds, while RRT-Connect [28] used a fixed arrival time of 3 seconds due to its inability to handle unknown arrival times. For RL methods, the Markovian reward uses $\beta_c = 5$, $\beta_g = 20$ and $\beta_l = 0$, and the non-Markovian reward uses $\beta_c = 10$, $\beta_g = 40$ and $\beta_l = 1$.

The learning curves in Fig. 7a demonstrate that MoRe-ERL achieves higher sample efficiency and yields better performance compared to ERL trained from scratch and step-based RL approaches, under both Markovian and non-Markovian reward settings. However, with a sufficient planning time budget, ST-RRT* achieves a success rate of 92.8% in this task. When the budget is reduced to 100 ms, the success rate of ST-RRT* drops to below 70%. In contrast, MoRe-ERL maintains a significantly faster inference time of 10.1 ms while achieving a competitive success rate of 88.9%. Note that the scenarios are initialized to be *adversarial* to residual methods. ST-RRT* does not suffer from such a disadvantage, which reduces the task complexity for ST-RRT*. A common solution is to wait for the boxes to settle and approach the goal. For a more complicated task where the environment is constantly moving, such as the dual-arm task, MoRe-ERL significantly outperforms ST-RRT*.

B. Dual-Arm

The dual-arm scenario involves a UR5 and a KUKA iiwa 14 robot, mounted on a shared workspace, as shown in Fig. 6b. The UR5 follows pre-defined trajectories, acting as a dynamic part of the environment, while the KUKA iiwa 14 actively avoids collisions and moves toward the goal pose.

Similar to the multi-box environment, episodes are randomly initialized and terminate when either the goal is reached or the pre-defined maximum duration of 5 seconds elapses. For step-based methods, the observation includes (a) the position and velocity of both robots, (b) the goal of the iiwa robot and (c) the current timestamp. As in V-A, the step-based residual method includes the next reference action in the observation, while MoRe-ERL incorporates 5 intermediate waypoints from the reference trajectories. Sampling-based baselines reported in TABLE II are given 5 seconds planning time budgets for each problem in a space-time state space $\mathbb{R}^{7+1}$. The maximum arrival time of ST-RRT* is set to 5 seconds, and RRT-Connect has a fixed arrival time. The Markovian reward is weighted by $\beta_c = 5$, $\beta_g = 20$ and $\beta_l = 0$ and the non-Markovian setting is same as V-A.

MoRe-ERL achieves a 76.7% success rate under the non-Markovian reward, outperforming ST-RRT*, which succeeds in only 39.2% of cases after 5 seconds of planning. It is worth mentioning that the test cases are randomly generated, and some of them may not be solvable. To approximate the upper bound of success rate, we increased the planning budget to 120 seconds, where ST-RRT* achieved success in 86% of the test cases. No significant performance gain is shown if the planning budget increases further. Fig. 7b shows that step-based methods failed to learn a reasonable policy in both Markovian and non-Markovian reward settings. MoRe-ERL's superior performance stems from its ability to identify the residual intervals while retaining critical behaviors, e.g., retracting from a shelf. These behaviors are usually difficult to learn from scratch.

Methods	**Multi-box**		**Dual-arm**	
	PT [s] ↓	Success ↑	PT [s] ↓	Success ↑
ST-RRT*	1.0	**0.928**	5.0	0.392
RRT-Connect	0.609	0.587	4.229	0.204
ERL + DMPs	0.011	0.092	0.010	0.002
ERL + DMPs + Residuals		0.584		0.168
BBRL	**0.010**	0.731	**0.0098**	0.133
Full Residual		0.881		0.674
Partial Replacement		0.812		0.299
MoRe-ERL (ours)		0.889		**0.767**

TABLE II: Results of MoRe-ERL and baseline methods in both environments, evaluated by planning time (PT) and success rate. The arrow ↓ indicates lower is better, and ↑ indicates higher is better. An episode is successful if the robot reaches the goal region (radius 0.1) without collisions. Sampling-based results are averaged over 10 seeds, RL results over 8 seeds.

C. Ablation Studies

Refinement Variants. We conducted ablation studies on different trajectory refinement strategies illustrated in Fig. 4. The results in Fig. 7a demonstrate that full residual achieves slightly better sample efficiency compared to MoRe-ERL in the multi-box task. This behavior can be attributed to the fact that full residual is a special case of MoRe-ERL with $\alpha_s = \tau$ and $\alpha_e = T$, leading to a reduced dimension of the searching space. However, in scenarios that require dexterous robot behaviors, such as the dual-arm task, identifying the residual interval using α_s and α_e shows clear advantages in both convergence speed and final performance. Meanwhile, partial replacement underperforms in both tasks, highlighting the effectiveness of learning motion residuals.

Movement Primitives. Prior work [29], [30] has modeled obstacles in the task space using potential fields, which are explicitly incorporated into the DMP formulation as coupling terms. However, these methods are limited to simple obstacle geometries, such as a single moving sphere, and therefore do not scale to our experimental setup, which involves complex static and dynamic structures like bookshelves and moving robot arms. To demonstrate the benefits of BMPs, we replace them with DMPs in both the learning-from-scratch (ERL + DMPs) and residual settings. Table II shows that BMPs consistently outperform DMPs, as DMPs achieve high collision-free rates but often fail to reach the goal.

D. Real-World Experiment

The hardware setup aligns with V-B, with modifications with respect to real-world calibration between two robots. Fig. 1 and Fig. 6c provide two distinct views of the setup. Initially, the policy is trained in simulation using the procedure outlined in V-B, achieving a success rate of approximately 95%. The trained policy is subsequently deployed on the real-world hardware. Episodes are designed such that the goal of the current episode becomes the starting point of the next, allowing for seamless sequential rollouts. The policy learned by MoRe-ERL demonstrates a minimal gap in transferring from simulation to the real world.

VI. Limitations and Future Works

We proposed MoRe-ERL, a general residual learning framework tailored for episodic reinforcement learning. MoRe-ERL can be built on top of any method within the ERL category and demonstrates significant improvements over learning from scratch. By identifying the crucial intervals within the reference trajectory and applying residual learning to these segments, MoRe-ERL enhances both learning efficiency and task performance. However, although MoRe-ERL is capable of refining the trajectories and achieved a high success rate, its performance highly depends on the quality of reference trajectories. In case of poor quality, MoRe-ERL has to make extra effort to escape from the pattern provided by the reference trajectory and can not demonstrate advantages. Furthermore, while MoRe-ERL offers a novel solution to refine trajectories, the timing to trigger and request such refinements is also crucial. An additional high-level decision-making layer is needed to trigger refinement and select best suited refinement strategies.

References

[1] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[2] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Advances in neural information processing systems*, vol. 21, 2008.

[3] F. Otto, H. Zhou, O. Celik, G. Li, R. Lioutikov, and G. Neumann, "Mp3: Movement primitive-based (re-) planning policy," *arXiv preprint arXiv:2306.12729*, 2023.

[4] F. Otto, O. Celik, H. Zhou, H. Ziesche, V. A. Ngo, and G. Neumann, "Deep black-box reinforcement learning with movement primitives," in *6th Annual Conference on Robot Learning (CoRL 2022)*, ser. Proceedings of Machine Learning Research, vol. 205. Machine Learning Research Press (ML Research Press), 2022, pp. 1244–1265.

[5] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, "Prodmp: A unified perspective on dynamic and probabilistic movement primitives," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2325–2332, 2023.

[6] W. Liao, G. Li, H. Zhou, R. Lioutikov, and G. Neumann, "Bmp: Bridging the gap between b-spline and movement primitives," *arXiv preprint arXiv:2411.10336*, 2024.

[7] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

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

[10] 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.

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

[12] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, "Residual policy learning," *arXiv preprint arXiv:1812.06298*, 2018.

[13] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[14] A. Ranjbar, N. A. Vien, H. Ziesche, J. Boedecker, and G. Neumann, "Residual feedback learning for contact-rich manipulation tasks with uncertainty," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2383–2390.

[15] J. Carvalho, D. Koert, M. Daniv, and J. Peters, "Adapting object-centric probabilistic movement primitives with residual reinforcement learning," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 405–412.

[16] 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.

[17] X. Huang, G. Sóti, H. Zhou, C. Ledermann, B. Hein, and T. Kröger, "Hiro: Heuristics informed robot online path planning using precomputed deterministic roadmaps," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8109–8116.

[18] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *Conference on Robot Learning*. PMLR, 2023, pp. 967–977.

[19] X. Huang, G. Sóti, C. Ledermann, B. Hein, and T. Kröger, "Planning with learned subgoals selected by temporal information," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9306–9312.

[20] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint, "St-rrt*: Asymptotically-optimal bidirectional motion planning through space-time," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3314–3320.

[21] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.

[22] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Berlin, Germany, 2013, pp. 1–10.

[23] D. Whitley, S. Dominic, R. Das, and C. W. Anderson, "Genetic reinforcement learning for neurocontrol problems," *Machine Learning*, vol. 13, pp. 259–284, 1993.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[25] P. Kicki, D. Tateo, P. Liu, J. Günster, J. Peters, and K. Walas, "Bridging the gap between learning-to-plan, motion primitives and safe reinforcement learning," in *8th Annual Conference on Robot Learning*, 2024.

[26] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-Spline Techniques*. Berlin, Heidelberg: Springer, 2002.

[27] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[28] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[29] H. Tan, E. Erdemir, K. Kawamura, and Q. Du, "A potential field method-based extension of the dynamic movement primitive algorithm for imitation learning with obstacle avoidance," in *2011 IEEE International Conference on Mechatronics and Automation*. IEEE, 2011, pp. 525–530.

[30] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Humanoids 2008-8th IEEE-RAS international conference on humanoid robots*. IEEE, 2008, pp. 91–98.