BMP: Bridging the Gap between B-Spline and Movement Primitives

Weiran Liao, Ge Li, Hongyi Zhou, Rudolf Lioutikov, Gerhard Neumann
Karlsruhe Institute of Technology
ge.li@kit.edu

Abstract: This work introduces **B-spline Movement Primitives (BMPs)**, a new Movement Primitive (MP) variant that leverages B-splines for motion representation. B-splines are a well-known concept in motion planning due to their ability to generate complex, smooth trajectories with only a few control points while satisfying boundary conditions, i.e., passing through a specified desired position with desired velocity. However, current usages of B-splines tend to ignore the higher-order statistics in trajectory distributions, which limits their usage in imitation learning (IL) and reinforcement learning (RL), where modeling trajectory distribution is essential. In contrast, MPs are commonly used in IL and RL for their capacity to capture trajectory likelihoods and correlations. However, MPs are constrained by their abilities to satisfy boundary conditions and usually need extra terms in learning objectives to satisfy velocity constraints. By reformulating B-splines as MPs, represented through basis functions and weight parameters, BMPs combine the strengths of both approaches, allowing B-splines to capture higher-order statistics while retaining their ability to satisfy boundary conditions. Empirical results in IL and RL demonstrate that BMPs broaden the applicability of B-splines in robot learning and offer greater expressiveness compared to existing MP variants.

Keywords: Movement Primitives, Planning, Robot Learning

1 Introduction

Movement Primitives (MPs)[1, 2, 3, 4] are a well-established concept in robot learning, offering a compact parameterization of movements that can be efficiently reused, adapted and combined to synthesize complex behavior. They can be categorized into dynamic system-based MPs and probabilistic-based MPs [4]. Dynamic system-based MPs, model movements as dynamic systems with an emphasis on ensuring global stability. The first of this kind is Dynamic Movement Primitives (DMP) [1]. DMP employs a second-order dynamic system that guarantees trajectory convergence to a goal attractor. While DMPs are effective for point-to-point tasks, they are less suited for more general motion tasks that require passing through several via points. Additionally, DMPs lack probabilistic modeling capabilities due to their dynamic system formulation, limiting their applicability in tasks that require trajectory distribution modeling, such as imitation learning (IL) and reinforcement learning (RL). On the other hand, probabilistic-based MPs model movements directly as trajectory distributions, allowing for flexibility in maintaining and adapting these distributions. A key example is Probabilistic Movement Primitives (ProMPs)[2], which represent trajectories as a linear combination of basis functions with weights. ProMPs use a linear Gaussian model, enabling probabilistic inference to handle complex interactions and adapt trajectories based on context. While ProMPs excel in modeling uncertainty and adapting to new situations, they struggle with satisfying boundary conditions like initial states, particularly in reinforcement learning tasks where robots are reset to different initial states at the beginning of each episode.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

In the realm of trajectory planning, B-splines are widely applied due to their minimal parameter requirements and local support of basis functions, which benefit optimization processes. In addition, B-splines can satisfy any boundary conditions, i.e., passing through specified positions with desired velocity, making them ideal for planning and replanning smooth trajectories. However, current usages of B-splines focusing on model single trajectory, ignoring the trajectory distributions [5, 6, 7, 8], limiting their usages in IL and RL where trajectories likelihood and temporal correlations are desired [4, 9].

In this paper, we propose BMPs, a novel approach that integrates B-splines as probabilistic-based Movement Primitives. This unified framework is suitable for both imitation learning (IL) and episodic reinforcement learning (ERL) [9, 10, 11, 12], retaining the strengths of MPs—such as probabilistic representation—while also ensuring the satisfaction of boundary conditions. Through three experiments, we demonstrate the effectiveness of BMPs in both IL and ERL settings, showcasing their broader applicability and expressiveness in robot learning.

2 Related Works

Dynamic Movement Primitives (DMPs) [1] model a trajectory as second-order dynamical systems (DS) with asymptotically vanishing nonlinear forcing input term. it is designed to ensure asymptotic stability, that all executing movements converge asymptotically to the goal attractor. Once a DMP is learned, to apply it to reproduce the movement adaptively as its original design purpose, it should be used in real-time integration form with real feedback values. Considering perturbations during movement execution, they are well-suited for point-to-point tasks, where the primary requirement is that the movement reaches a desired goal state. Many tasks, however, require the robot to dynamically follow desired trajectories but not just to converge to a fixed point, i.e. the shape of the movement after perturbations is essential for the tasks. There are various modification of DMP for different applications summarized in [13].

In imitation learning (IL), through setting the initial condition and goal atrractor of the DS, DMP has the advantage of adapting learned movement to new start positions, velocity, and new end positions while keeping the learned movement shape. In RL or general planning tasks, DMP ensures a precise smooth start. Notably, The forcing term containing the parameters shaping the trajectory is the input to the dynamic system. This makes the learning process need to propagate through the dynamic system and, hence makes DMPs unable to capture trajectory statistics.

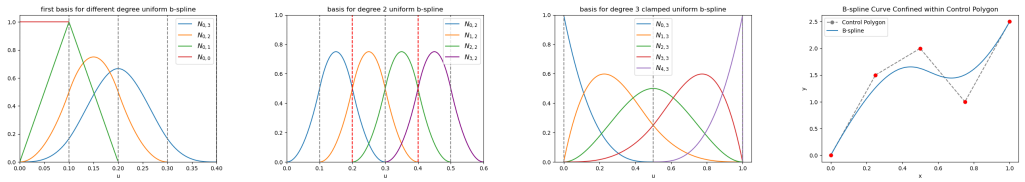
Probabilistic Movement Primitives (ProMPs)[2] directly represent a trajectory by a linear combination of basis functions with weights and thus utilize a linear Gaussian model to model the trajectory distribution. A single Degree of Freedom (DoF) example can be written as

$$\boldsymbol{y}_{0:T} = \boldsymbol{\Phi}_{0:T}^{\top}\boldsymbol{\omega} + \boldsymbol{\epsilon}_{\boldsymbol{y}}, \tag{1}$$

$$p(\boldsymbol{y}_{0:T}|\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}) = \mathcal{N}(\boldsymbol{y}_{0:T}|\boldsymbol{\Phi}_{0:T}^{\top}\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Phi}_{0:T}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{\omega}}\boldsymbol{\Phi}_{0:T} + \boldsymbol{I}\sigma_y^2), \tag{2}$$

where $\boldsymbol{y}_{0:T}$ is the desired trajectory evaluated at time steps from 0 to T. $\boldsymbol{\Phi}_{0:T}$ defines the $n \times T$ dimensional basis matrix consisting of evaluated values at all time steps of n normalized radial basis function (RBF). $\epsilon_y \sim (0, \sigma_y)$ is a zero-mean i.i.d. Gaussian noise. $\boldsymbol{\omega}$ defines an n-dimensional parameters vector that fellows a Gaussian distribution $\mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$. The (2) can be extended to model multiple DoF trajectories as

$$\begin{aligned} p(\boldsymbol{Y}_{0:T}|\boldsymbol{\mu}_{\boldsymbol{\Omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\Omega}}) = \mathcal{N}\Bigg(\begin{bmatrix} \boldsymbol{y}_{1,0:T} \\ \vdots \\ \boldsymbol{y}_{d,0:T} \end{bmatrix} \Bigg| \begin{bmatrix} \boldsymbol{\Phi}_{0:T}^{T} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{\Phi}_{0:T}^{T} \end{bmatrix} \boldsymbol{\mu}_{\boldsymbol{\Omega}}, \\ \begin{bmatrix} \boldsymbol{\Phi}_{0:T}^{T} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{\Phi}_{0:T}^{T} \end{bmatrix} \boldsymbol{\Sigma}_{\boldsymbol{\Omega}} \begin{bmatrix} \boldsymbol{\Phi}_{0:T}^{T} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{\Phi}_{0:T}^{T} \end{bmatrix}^{\top} + \boldsymbol{\Sigma}_y \Bigg) \end{aligned} \tag{3}$$



(a) (b) (c) (d)

Figure 1: (a) The first basis for B-spline of degree 0 1 2 3. (b) Basis functions for a degree 2 uniform B-spline with 4 basis defined on knot vector $[0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6]$, the valid spans for representing a trajectory are $[0.2,\ 0.4]$, where there are always 3 non-zero bases in each span. (c) Clamped uniform B-spline of degree 3 with 5 Basis defined on knot vector $[0,\ 0,\ 0,\ 0,\ 0.5,\ 1,\ 1,\ 1,\ 1]$. (d)The Clamped B-spline trajectory starts exactly from first control point and ends at the last control point, the whole trajectory stays inside the polygon region of control points

where d indicates the DoF, $\boldsymbol{\mu}_{\boldsymbol{\Omega}} = [\boldsymbol{\mu}_{\boldsymbol{\omega}_1}^T, \ldots, \boldsymbol{\mu}_{\boldsymbol{\omega}_d}^T]^T$. $\boldsymbol{\Sigma}_{\boldsymbol{\Omega}}$ is a $(d \times n) \times (d \times n)$ multi-DoF weights covariance matrix capturing also the correlation across DoFs. By maintaining the trajectory distributions, ProMPs can generate trajectories conditioned on observed trajectory values. However, ProMP is unable to satisfy precise boundary conditions, because the normalized RBF function never reaches exactly zero value.

Probabilistic Dynamic Movement Primitives (ProDMPs) [4] is proposed to provide a unified framework that retains the benefits of both ProMP and DMP, combining the flexibility of probabilistic modeling with the ability to satisfy an initial condition. It solves the differential equation of DMP to construct a set of basis functions in position space, thus trajectories can be represented in a similar form to ProMP,

$$y(t) = c_1 y_1(t) + c_2 y2(t) + \boldsymbol{\Phi}_{prodmp}^T(t)\boldsymbol{\omega}_g, \tag{4}$$

where $c_1 y_1(t) + c_2 y2(t)$ ensures satisfying initial position and velocity conditions. Although it is only used as a pure trajectory generator like ProMP in the ProDMP paper, the learned weights can still applied to a DMP in adaptive manner, since all basis functions are directly solved from the DMP formulation. However, the ProDMP can only satisfy initial conditions. In addition, the basis functions $\boldsymbol{\Phi}_{prodmp}$ exhibit substantial magnitude differences among basis, which need to be rescaled in RL practice. And the basis functions overlap with each other throughout the whole movement duration, which is undesired since changing one weight will affect the whole trajectory.

3 Bridging the Gap between B-Spline and Movement Primitives

B-spline [14] is a piecewise polynomial function commonly used for approximating curves and surfaces. For a B-spline of degree p (where $p \geq 0$), the B-spline curve $y(u)$ is expressed as a linear combination of basis functions:

$$y(u) = \sum_{i=0}^{n-1} N_{i,p}(u)\, c_i, \quad 0 < p < n, \quad u \in [u_0, u_m]. \tag{5}$$

where c_i are the control points, the weights for corresponding basis functions. $N_{i,p}(u)$ are the B-spline basis functions of degree p, defined on the knot vector $\boldsymbol{u} = [u_0, ..., u_m]$, with $u_i \leq u_{i+1}$. u is a generalized time variable within the interval $[u_0, u_m]$. The basis functions $N_{i,p}(u)$ are defined recursively using Cox–de Boor recursion formula [14]. For degree $p = 0$, the basis functions are piecewise constant:

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1}, \text{ if } u_i \leq u \leq u_{i+1} \text{ for } i = n-1, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

For degree $p > 0$, the recursive definition is given by:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u), \tag{7}$$

For example, as shown in 1a, the degree 0 basis function $N_{0,0}$ is a step function valid in the knot span $[u_0, u_1)$. The degree 1 basis function $N_{1,0}$ is a triangle spanning $[u_0, u_2)$, with its peak at at u_1. The degree 2 basis function $N_{2,0}$ forms a parabola over $[u_0, u_3)$.

In each valid knot span for modeling, there are $p+1$ non-zero basis functions. For a degree 2 B-spline, for instance in 1b, the first valid span is $[u_2, u_3)$, and the last is $[u_{m-3}, u_{m-2}]$. Generally, for a degree p B-spline with n basis, the knot vector $\boldsymbol{u} = [u_0, ..., u_m]$ contains $m+1$ knots, satisfying the condition $m = n + p$. The valid spans are $[u_p, ..., u_{m-p}]$, and within each span $[u_i, u_{i+1})$, the curve is influenced by $p+1$ control points $c_{i-p}, ..., c_i$. An important property of the B-spline is the local support of the basis functions—each basis function only affects a limited portion of the curve, providing local control. Additionally, due to the recursive definition, the derivative of a B-spline curve of degree p with n control point is a B-spline of degree $p-1$ with $n-1$ control points, allowing easy derivation of the control points and basis functions of the derivative. Finally, B-splines possess the convex hull property, meaning the curve is contained within the convex hull of its control points shown in 1d. This property is frequently used to efficiently impose bound constraints.

We adopt the clamped uniform B-spline as MP. Uniform means all knots in the valid spans are equidistantly located. Clamped means that the curve passes through the first and last control points as shown in 1d, which is utilized to satisfy boundary conditions. This is achieved by repeating both the first and last knot $p+1$ times, as demonstrated in 1c, ensuring that the first basis function evaluates to 1 at time point $u = 0$ while the other basis functions are evaluated to 0, and the same applies reversely at the end time point.

BMP: Using B-spline to extend ProMP. For a single DoF trajectory, the B-spline can be write in a similar form to ProMP as

$$\boldsymbol{y}_{0,...T} = [\boldsymbol{N}_{0,p}(u_{0,...,T}) \; ... \; \boldsymbol{N}_{n-1,p}(u_{0,...,T})] \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = \boldsymbol{\Phi}^T(u_{0,...,1}) \, \boldsymbol{c} \tag{8}$$

where $y_{0,...T}$ are the trajectory values in the time steps $[0, ..., T]$, the $u_{0,...,T}$ are linear phases values with $u(t) = \frac{t-t_0}{T}$ ranging from 0 to 1. The B-spline basis function $N_{i,p}(u)$ is defined on a knot vector $\boldsymbol{u} = [u_0, u_1, ..., u_p, ..., u_{m-p}, ..., u_{m-1}, u_m]$, where $u_0, u_1, ..., u_p = 0$, $u_{m-p}, ..., u_{m-1}, u_m = 1$, and $u_{i+1} - u_i = \Delta = \frac{1}{m-2p}$ for $i \in [p, ..., m-p-1]$.

Utilizing the derivative property of the B-spline, we have the trajectory velocity with respect to phase in the form of

$$\dot{\boldsymbol{y}}_{0,...T} = [\boldsymbol{N}_{1,p-1}(u_{0,...,T}) \; ... \; \boldsymbol{N}_{n-1,p-1}(u_{0,...,T})] \begin{bmatrix} c_0^{(1)} \\ \vdots \\ c_{n-2}^{(1)} \end{bmatrix} = \dot{\boldsymbol{\Phi}}^T(u_{0,...,1}) \, \boldsymbol{c}^{(1)}, \tag{9}$$

where the velocity control point is computed through

$$c_i^{(1)} = \frac{p}{\Delta}(c_{i+1} - c_i). \tag{10}$$

This can be recursively applied to obtain the acceleration profile. The trajectory starts exactly from the first control point c_0 and ends at the last control point n_{n-1}. The velocity profile starts exactly from the first velocity control point c_0^1 and ends at the last velocity control point c_{n-2}^1. In the above derivation, the velocity are computed with respect to the phase but not the time. It needs to be divided by the duration T to get real velocity.

Depending on the initial conditions and end conditions we are about to impose, we can prescribe values to the first control points and the last control points. To impose initial position y_0 and initial velocity $\dot{y}_0$, we solve the linear equations

$$\begin{cases} c_0 = y_0 \\ c_0^{(1)} = \frac{p}{\Delta}(c_1 - c_0) = \dot{y}_0 \end{cases} \Rightarrow \begin{cases} c_0 = y_0 \\ c_1 = \frac{\Delta}{p}\dot{y}_0 + c_0 \end{cases}, \tag{11}$$

and set the first two control points respectively. The same applies reversely for imposing end position or end position plus end velocity. Sometimes, only the end velocity but no end position is to be prescribed, for example, when a zero end velocity is desired but the position should be learned. It can be solved by

$$c_{n-2}^{(1)} = \frac{p}{\Delta}(c_{n-1} - c_{n-2}) = \dot{y}_e \;\Rightarrow\; c_{n-2} = -\frac{\Delta}{p}\dot{y}_e + c_{n-1}. \tag{12}$$

where $\dot{y}_e$ is the desired end velocity. This means the second last control point c_{n-2} is set as a pre-given constant $-\frac{\Delta}{p}\dot{y}_e$ plus the later learned last control point value c_{n-1}. It is obvious that the to-be-learned weights are these control points in between and maybe also the last control point.

Due to the linear combination representation, the linear Gaussian modeling in ProMP can be applied to BMP. For instance, in the case of given initial position and velocity conditions, end position and velocity conditions, the single DoF trajectory distribution takes a similar form to (2):

$$p(\boldsymbol{y}_{0:T}|\boldsymbol{\mu}_{\boldsymbol{c}_{2:n-3}}, \boldsymbol{\Sigma}_{\boldsymbol{c}_{2:n-3}}) = \mathcal{N}(\boldsymbol{y}_{0:T}|\boldsymbol{\Phi}_{0:T,[2:n-3]}^{\top}\boldsymbol{\mu}_{\boldsymbol{c}_{2:n-3}} + \boldsymbol{d}, \boldsymbol{\Phi}_{0:T,[2:n-3]}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{c}_{2:n-3}}\boldsymbol{\Phi}_{0:T,[2:n-3]} + \boldsymbol{I}\sigma_y^2), \tag{13}$$

where $\boldsymbol{d}$ is the deterministic part of the trajectory, defined by boundary conditions. This can be extended to the multi-DoF case to model the uncertainty and correlation across both dimension and time, following the same way as in (3).

In the practical implementation, the trajectory is composed of a constant line with initial position value y_0 and the linear combination of basis functions with control points, i.d. the B-spline only learns the residual based on the initial position and the initial position feed to B-spline is always 0. This makes the variance of the to-be-learned control points smaller, which significantly improves the learning performance compared to simply using the B-spline. Additionally, the constant line can be also replaced by a linear trajectory pointing from the initial position to the goal position.

4 Experiments

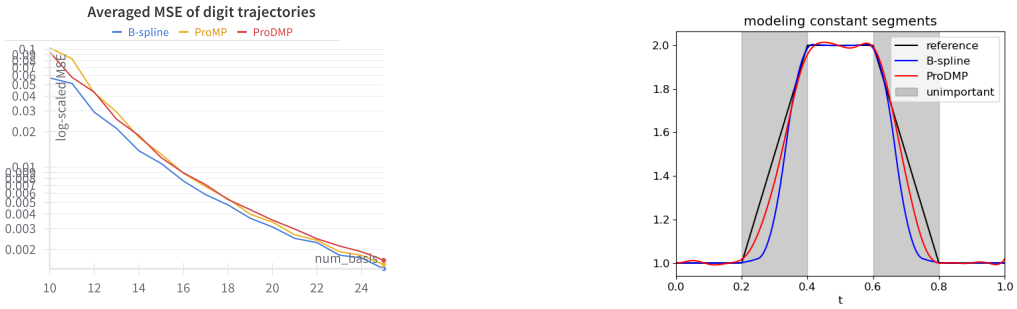
4.1 B-spline versus MPs

We first show that B-splines are generally more expressive than ProDMP and ProMP. As illustrated in 2a, by regressing B-splines and MPs on a set of digit trajectories, B-splines achieve a lower mean squared error (MSE) loss. Additionally, an effective trajectory representation should be capable of representing constant segments, corresponding to a robot remaining at specific configurations. For a trajectory representation that is a linear combination of basis functions, this requirement is satisfied by normalized basis functions, as is the case for both B-spline and ProMP. However, as depicted in 2b, ProDMP is unable to represent constant segments.

B-spline offers significant advantages as a trajectory generator for planning and RL tasks, particularly in their ability to specify arbitrary boundary conditions. In contrast, ProDMP guarantees only the satisfaction of initial position and velocity, while ProMP does not support the specification of either initial or final conditions. As demonstrated in 3, a goal-reaching toy task, where a trajectory must be generated from given initial conditions to reach a target with a desired velocity while avoiding obstacles. In addition, the trajectory should satisfy a given velocity bound. By setting boundary conditions, B-splines complete the task within a few iterations, where it only learns to avoid obstacles. Besides, utilizing the convex hull property of the B-spline, we can simply set soft constraints for velocity control points to satisfy velocity bounds. In contrast, accommodating above requirements through reward shaping in ProDMP is untrivial.

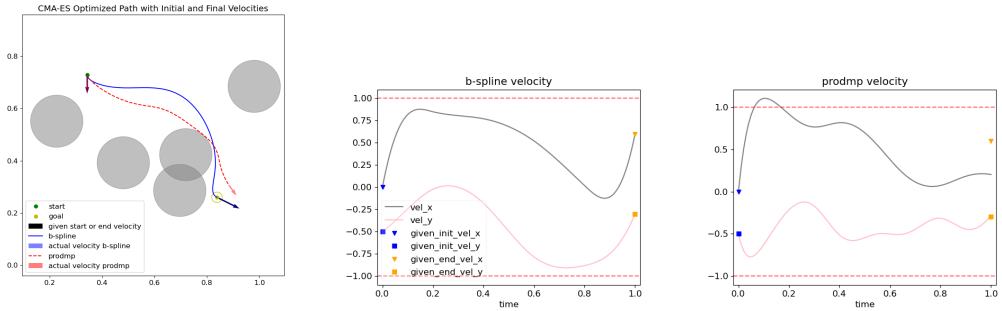
4.2 BMP for Imitation Learning and Reinforcement Learning

Imitation Learning We embed BMP into a neural network as a generative model, to generate digit-writing trajectories conditioned on digit images from the synthetic-MNIST dataset [16]. We adopt an encoder-decoder neural network architecture and its training procedure proposed in [4], where the neural network takes a set of digit writing images as input and predict the mean $\boldsymbol{\mu}_{\boldsymbol{w}}$ and the



(a) (b)

Figure 2: (a) The log-scaled averaged regression MSE loss on 20000 3-second digit-writing trajectories, by applying B-spline and MPs with different numbers of basis functions. (b) Regressing B-spline and ProDMP on a trajectory with three constant segments, where the grey shadowed transition segments are unimportant and not considered in regression. ProDMP exhibits obvious wiggles in the all constant segments $[0,\ 0.2],\ [0.4,\ 0.6],\ [0.8,\ 1]$



(a) (b) (c)

Figure 3: Using CMA-ES [15] algorithm for 50 iterations to generate trajectories from given initial states to reach goal states while avoiding obstacles. (a) Generated best trajectory path within 50 iterations. (b),(c) The velocity profile of the B-spline and ProDMP trajectory, where the red dash lines are the velocity bounds.

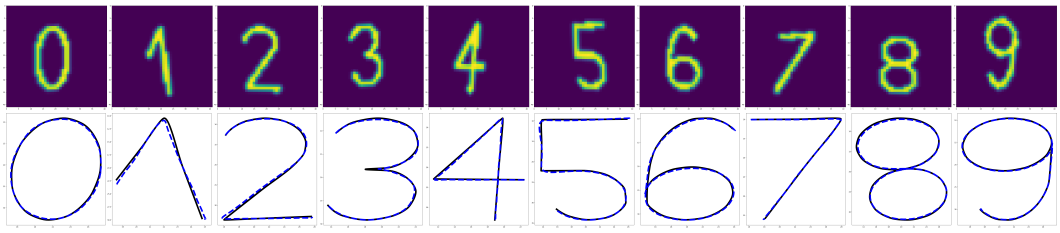
Cholesky decomposition $\boldsymbol{L_w}$ of the covariance $\boldsymbol{\Sigma_w}$ of the weights distribution of the BMP. Then BMP maps this weights distribution to the trajectory distribution $\mathcal{N}(\boldsymbol{\mu_\Lambda}, \boldsymbol{\Sigma_\Lambda})$.

Since we maintain a distribution of trajectories through BMP, we can leverage probabilistic modeling techniques to train this generative model. Specifically, we aim to minimize the negative log-likelihood of the ground truth trajectories on the predicted trajectory distribution, i.e., $-\log \mathcal{N}(\boldsymbol{\Lambda}|\boldsymbol{\mu_\Lambda}, \boldsymbol{\Sigma_\Lambda})$. Here, $\boldsymbol{\Lambda} = \{\boldsymbol{y}_t\}_{t=0..T}$ is the trajectory ground truth, and $\boldsymbol{\mu_\Lambda}, \boldsymbol{\Sigma_\Lambda}$ the mean and covariance of the predicted trajectory distribution conditioned on a set of image observations $\mathcal{O}$. However, directly computing the log-likelihood of that a high-dimensional Gaussian distribution during each training step is computationally infeasible due to the need to invert the $TD \times TD$ dimension covariance matrix $\boldsymbol{\Sigma_\Lambda}$. To get around this problem while keep capturing the temporal correlations in trajectories, J time point pairs $\{(t, t')_j\}_{j=1,\dots,J}$ are randomly sampled and for each time point pair a log-likelihood is calculated, where only J times $2D \times 2D$ matrix inversion is performed, resulting in an alternative loss function

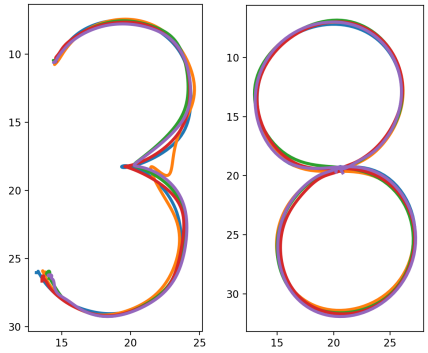
$$\mathcal{L}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda} \mid \mathcal{O}) = -\frac{1}{J} \sum_{j=1}^{J} \log \mathcal{N}\left(\boldsymbol{y}_{(t,t')_j} \mid \boldsymbol{\mu}_{(t,t')_j}, \boldsymbol{\Sigma}_{(t,t')_j}\right). \tag{14}$$

Although this training objective is biased from the exact negative log-likelihood of trajectories on the predicted correlated trajectory distribution, it captures correlations and reconstructs high-quality

trajectories as shown in 4. Using BMP and this training procedure allows training a generative model to sample trajectories.

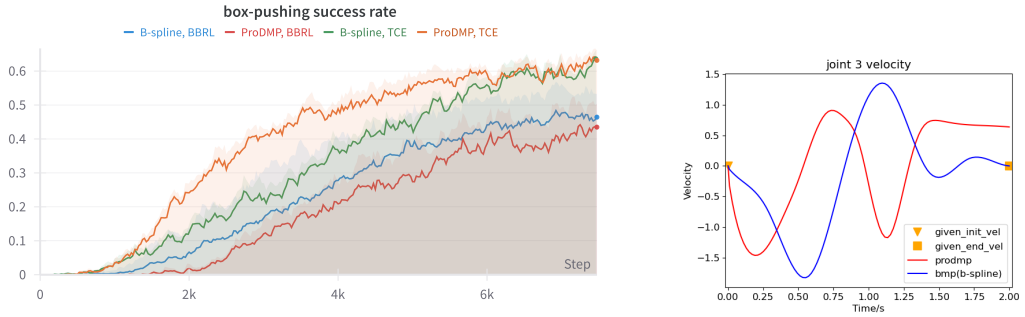


(a)



(b)

Figure 4: (a) A batch of digit images and reconstructed trajectories using BMP. (b) Sampling trajectories from predicted distribution conditioned on an image of digit '3' and '8'.



(a) (b)

Figure 5: (a) The averaged success rate of 4 random seeds. Both BMP (B-spline) and ProDMP use in total 9 basis functions. The basis functions of ProDMP are scaled into the same value range to make it work. (b) The joint velocity profile of joint 3, where the velocity at 0s and 2s should be 0.

Episodic Reinforcement Learning (ERL) treats RL problem as a contextual black-box optimization problem, where the action sequence for an episode is parameterized by $\boldsymbol{\omega}$ and the whole policy roll-out process is considered as one entity, a black-box. It aims to maximize the expected return of an episode policy roll-out $R(\boldsymbol{w}, \boldsymbol{c})$ by optimizing the contextual search distribution $\pi(\boldsymbol{\omega}|\boldsymbol{c})$ of the parameters ω, i.d.

$$\arg\max_{\pi(\boldsymbol{w}|\boldsymbol{c})} \mathbb{E}_{p(c)}\left[\mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{c})}[R(\boldsymbol{w}, \boldsymbol{c})]\right], \tag{15}$$

where $\boldsymbol{c}$ is the context and $p(\boldsymbol{c})$ is the distribution of the context. Compared to step-based RL, the underlying world model is not necessarily a Markov Decision Process (MDP). Combining with MP,

ERL can generate smoother trajectories and allows consistent oriented exploration. We adopt two ERL algorithms:

Deep-Black Box Reinforcement Learning (BBRL) [10], whose surrogate learning objective is

$$\hat{J}(\pi_{\boldsymbol{\theta}}, \pi_{\boldsymbol{\theta}_{\text{old}}}) = \mathbb{E}_{(\boldsymbol{c},\boldsymbol{w})\sim p(\boldsymbol{c}),\pi_{\boldsymbol{\theta}_{\text{old}}}} \left[\frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{w}|\boldsymbol{c})}{\pi_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{w}|\boldsymbol{c})} A^{\pi_{\boldsymbol{\theta}_{\text{old}}}}(\boldsymbol{c}, \boldsymbol{w}) \right], \tag{16}$$

where $A^{\pi_{\boldsymbol{\theta}_{\text{old}}}}(\boldsymbol{c}, \boldsymbol{w})$ is an advantage function. The objective is to maximize the advantage-weighted likelihood at the parameter level. BBRL can be simply applied to the B-spline.

Temporal Correlated Exploration (TCE) [9] divides the whole trajectory into K segments $[\boldsymbol{y}_t]_{t=t_k:t'_k}$. By maintaining trajectory distribution and taking the same philosophy in (14), the learning objective takes the form

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\text{old}}} \left[\frac{1}{K} \sum_{k=1}^{K} \frac{p_{\pi_{\text{new}}}\left([\mathbf{y}_t]_{t=t_k:t'_k} \middle| \mathbf{s} \right)}{p_{\pi_{\text{old}}}\left([\mathbf{y}_t]_{t=t_k:t'_k} \middle| \mathbf{s} \right)} A^{\pi_{\text{old}}}\left(s_{t_k}, [\mathbf{y}_t]_{t=t_k:t'_k} \right) \right]. \tag{17}$$

In contrast to BBRL, TCE maximizes the advantage-weighted likelihood for each segment at the trajectory level, which allows a finer-grained update of the policy while using the same roll-out data as in BBRL. This training scheme requires trajectory distribution, thus can only apply to BMP.

We demonstrate using BMP within both algorithms in a contact-rich manipulation task, the box-pushing task, where the objective is to move a box to a specified goal position with a specified goal orientation using a rod fixed on the end effector of a seven DoF Franka Emika Panda robot. The result is evaluated in terms of success rate, as given in 5a. With the BBRL algorithms, the B-spline outperforms the ProDMP, possibly because the locally supported basis function benefits the optimization process. By maintaining trajectory distribution and using TCE, the convergence rate and final success rate improve significantly. The ProDMP shows faster growth in TCE at the early stage, we hypothesize that this is because ProDMP has a goal basis spanning the whole trajectory duration, which can enforce a temporal correlation in the trajectory distribution at the early stage. This may benefit the TCE training scheme. Generally, BMP shows a comparable performance to ProDMP and ensures satisfying prescribed end conditions, but ProDMP is unable to ensure end velocity conditions, as shown in 5b.

In summary, B-spline is generally a better trajectory representation than previous MPs. Utilizing B-spline as a probabilistic MP, the BMP, enables the probabilistic modeling for B-spline and enhances its application in IL and RL.

5 Conclusion

Our work presents using B-spline as MP in the ProMP framework, the BMP. This allows using probabilistic techniques for learning and ensuring satisfying boundary conditions. BMP can be a unified MP framework for imitation learning and reinforcement learning. In future work, we will further investigate the possible advantages of BMP. Especially, we are investigating how to efficiently and effectively utilize the convex hull property of B-spline to impose kinodynamic constraints in BBRL and TCE algorithms for practical reinforcement learning tasks.

Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

[1] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013. doi:10.1162/NECO_a_00393.

[2] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann. Probabilistic movement primitives. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/e53a0a2978c28872a4505bdb51db06dc-Paper.pdf.

[3] Y. Zhou, J. Gao, and T. Asfour. Learning via-point movement primitives with inter-and extrapolation capabilities. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4301–4308. IEEE, 2019.

[4] 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*, page 1–8, 2023. ISSN 2377-3766. doi:10.1109/LRA.2023.3248443.

[5] V. Usenko, L. Von Stumberg, A. Pangercic, and D. Cremers. Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 215–222, Vancouver, BC, Sept. 2017. IEEE. ISBN 978-1-5386-2682-5. doi:10.1109/IROS.2017.8202160. URL http://ieeexplore.ieee.org/document/8202160/.

[6] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen. Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight. *IEEE Robotics and Automation Letters*, 4(4): 3529–3536, Oct. 2019. ISSN 2377-3766. doi:10.1109/LRA.2019.2927938. URL https://ieeexplore.ieee.org/document/8758904. Conference Name: IEEE Robotics and Automation Letters.

[7] P. Kicki, P. Liu, D. Tateo, H. Bou-Ammar, K. Walas, P. Skrzypczyński, and J. Peters. Fast Kinodynamic Planning on the Constraint Manifold with Deep Neural Networks, Jan. 2023. URL http://arxiv.org/abs/2301.04330. arXiv:2301.04330 [cs].

[8] P. Kicki, D. Tateo, P. Liu, J. Guenster, J. Peters, and K. Walas. Bridging the gap between learning-to-plan, motion primitives and safe reinforcement learning, 2024. URL https://arxiv.org/abs/2408.14063.

[9] G. Li, H. Zhou, D. Roth, S. Thilges, F. Otto, R. Lioutikov, and G. Neumann. Open the black box: Step-based policy updates for temporally-correlated episodic reinforcement learning. In *ICLR 2024 : The Twelfth International Conference on Learning Representations, Vienna, 7th-11th May 2024*. International Conference on Learning Representations, ICLR, 2024.

[10] 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)*, volume 205 of *Proceedings of Machine Learning Research*, pages 1244–1265. Machine Learning Research Press (ML Research Press), 2022.

[11] G. Li, D. Tian, H. Zhou, X. Jiang, R. Lioutikov, and G. Neumann. Top-erl: Transformer-based off-policy episodic reinforcement learning, 2024. URL https://arxiv.org/abs/2410.09536.

[12] F. Otto, H. Zhou, O. Celik, G. Li, R. Lioutikov, and G. Neumann. Mp3: Movement primitive-based (re-)planning policy, 2023. URL https://arxiv.org/abs/2306.12729.

[13] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel. Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13): 1133–1184, 2023. doi:10.1177/02783649231201196. URL https://doi.org/10.1177/02783649231201196.

[14] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, Berlin, Heidelberg, 2002. ISBN 978-3-642-05240-8. doi:10.1007/978-3-662-04947-3.

[15] N. Hansen. The CMA Evolution Strategy: A Tutorial, Mar. 2023. URL http://arxiv.org/abs/1604.00772. arXiv:1604.00772 [cs, stat].

[16] R. Pahič, B. Ridge, A. Gams, J. Morimoto, and A. Ude. Training of deep neural networks for the generation of dynamic movement primitives. *Neural Networks*, 127:121–131, 2020. ISSN 0893-6080. doi:https://doi.org/10.1016/j.neunet.2020.04.010. URL https://www.sciencedirect.com/science/article/pii/S0893608020301301.