ProDMP: A Unified Perspective on Dynamic and Probabilistic Movement Primitives

Ge Li, Zeqi Jin, Michael Volpp, Fabian Otto, Rudolf Lioutikov, Member, IEEE, and Gerhard Neumann

Abstract—Movement Primitives (MPs) are a well-known concept to represent and generate modular trajectories. MPs can be broadly categorized into two types: (a) dynamics-based approaches that generate smooth trajectories from any initial state, e.g., Dynamic Movement Primitives (DMPs), and (b) probabilistic approaches that capture higher-order statistics of the motion, e.g., Probabilistic Movement Primitives (ProMPs). To date, however, there is no MP method that unifies both, i.e., that can generate smooth trajectories from an arbitrary initial state while capturing higher-order statistics. In this letter, we introduce a unified perspective of both approaches by solving the ODE underlying the DMPs. We convert expensive online numerical integration of DMPs into position and velocity basis functions that can be used to represent trajectories or trajectory distributions similar to ProMPs while maintaining all the properties of dynamical systems. Since we inherit the properties of both methodologies, we call our proposed model Probabilistic Dynamic Movement Primitives (ProDMPs). Additionally, we embed ProDMPs in deep neural network architecture and propose a new cost function for efficient end-to-end learning of higher-order trajectory statistics. To this end, we leverage Bayesian Aggregation for non-linear iterative conditioning on sensory inputs. Our proposed model achieves smooth trajectory generation, goal-attractor convergence, correlation analysis, nonlinear conditioning, and online re-planning in one framework.

Index Terms—Movement primitives, imitation learning, trajectory covariance learning, probabilistic learning.

I. INTRODUCTION

MOVEMENT Primitives Movement Primitive (MP) are a prominent tool for motion representation and synthesis in robotics. They serve as basic movement elements, modulate the motion behavior, and form more complex Movements through combination or concatenation. This work focuses on trajectory-based movement representations [1], [2]. Given a parameter vector, such representations generate desired trajectories for the robot to follow. These methods have gained much popularity in imitation and reinforcement learning (IL, RL) [3], [4], [5], [6], [7] due to their concise parametrization and flexibility. Current methods can be roughly classified into approaches based on dynamical systems [1], [8], [9], [10], [11] and probabilistic approaches [2], [12], [13], with both types offering their own advantages. The dynamical systems-based approaches, such as Dynamic Movement Primitives (DMPs), guarantee that the generated trajectories start precisely at the current position and velocity of the robot, which allows for smooth trajectory replanning i.e., changing the parameters of the MP during motion execution [11], [14]. However, since DMPs represent the trajectory via the forcing term instead of a direct representation of the position, numerical integration from acceleration to position has to be applied iteratively, which constitutes an additional workload and makes the trajectory statistics estimation difficult [15]. Probabilistic methods, such as Probabilistic Movement Primitives (ProMPs), are able to acquire such statistics, thus making them the key enablers for acquiring variable-stiffness controllers and the trajectory’s temporal and Degree of Freedom (DoF) correlation. These methods further perform as generative models, facilitating the sampling of new trajectories. However, the lack of internal dynamics of these approaches suffers from discontinuities in position and velocity between old and new trajectories in the case of replanning.

In this work, we propose Probabilistic Dynamic Movement Primitive (ProDMP) which unify both methodologies. We show that the trajectory of a DMPs, obtained by integrating its second-order dynamical system, can be expressed by a linear basis function model that depends on the parameters of the DMPs, i.e., the weights of the forcing function and the goal attractor. The linear basis functions can be obtained by integrating the original basis functions used in the DMPs - an operation that only needs to be performed once offline in the ProDMP. Recently, MP research has been extended to deep neural network (NN) architectures [10], [11],[13] that enable conditioning the trajectory on high-dimensional inputs, e.g., images. Following these ideas, we embed ProDMP into a deep architecture that allows non-linear conditioning on a varying number of conditioning events. These events are aggregated using Bayesian Aggregation (BA) into a latent probabilistic representation [16] which is mapped to a Gaussian distribution in the parameter space of the ProDMP. We summarize the contributions of this letter as:

1) We unify ProMPs and DMPs into one consistent framework that inherits the benefits of both.
2) We enable to compute distributions and to capture correlations of DMPs trajectories, while
3) the robot’s current state can be inscribed into the trajectory distribution through initial conditions, allowing for smooth replanning.

4) Moreover, the offline computation of the position and velocity basis functions significantly reduced the workload of NN architectures and decreased the computation time by a factor of 10.

5) Hence, we embed ProDMP in a deep encoder-decoder architecture that allows non-linear conditioning on a set of high-dimensional observations with varying information levels.

We evaluate our method on three digit-writing tasks using images as inputs, a simulated robot pushing task with a complex physical interaction, and a real robot picking task with shifting object positions. We compare our model with state-of-the-art NN-based DMPs [9], [10], [11] and the NN-based probabilistic method [13].

II. RELATED WORK

Paraschos et al. [2] established ProMPs to model MP as a trajectory distribution that captures temporal and DoF correlation. ProMPs use a Gaussian distribution over the parameters and model it to the corresponding trajectory distribution using a linear basis function model. In contrast, such a mapping is not allowed for the DMPs-based approaches. Previous methods, like GMM/GMR-DMPs [17], [18] used Gaussian Mixture Models to cover the trajectories’ domain. Yet, this does not capture temporal correlation nor provide a generative approach to trajectories. Other methods which have learned distributions over DMPs weights [19], [20], do not connect the weights distribution to the trajectory distribution, as trajectories can only be obtained by integration. Hence, it is also hard to learn the weights distribution reversely from trajectories in an end-to-end manner. Methods like [21], [22] model the motion’s statistics using a state-dependent dynamic system and directly learn the parameters of the governing differential equation from the trajectory. However, these methods have difficulties in modeling highly-dynamic time-dependent motions and correlations as well as scaling the execution speed of these motions. Moreover, it is not straightforward to integrate high-dimensional sensory observations in these architectures. To learn DMPs parameters from high-dimensional sensory inputs, [9], [10] designed an encoder-decoder architecture to learn a single DMPs from digit images, and derive the gradient of the trajectory with respect to the learnable parameters. [11] propose Neural Dynamic Policies (NDP) that allow replanning the DMPs parameters throughout the trajectory execution in both IL and RL settings. The learning objective of these two methods for IL is to optimize the mean squared error (MSE) between the predicted and the ground-truth trajectories using back-propagation. However, to formulate a trajectory, DMPs must apply numerical integration in an iterative manner during the NN training [9], [10], [11], which significantly increases the computational workload, rendering these approaches cumbersome to use. Additionally, the integration-based trajectory representation limits the use of probabilistic methods and hence these NN-DMPs approaches cannot be trained using a log-likelihood (LL) loss.

Probabilistic MP methods have also been extended with deep architectures. [13] directly use a Conditional Neural Processes (CNP) model [23] as a trajectory generator, i.e., Conditional Neural Movement Primitive (CNMP), to predict the trajectory distribution. While such a model enables non-linear conditioning on high-dimensional inputs, it can only predict an isotropic trajectory variance at each time step. The temporal and DoF correlations are missing, which makes sampling consistently in time and DoF infeasible. Besides, both ProMPs and CNMP neglect dynamics, i.e., when changing trajectory parameters in run-time, the newly generated trajectory will jump at the replanning time point. To execute such trajectories, a heuristic controller is used to freeze the time and catch up with the jump [13]. However, such a waiting mechanism does not scale to time-sensitive motions and tasks.

III. A UNIFIED PERSPECTIVE ON DYNAMIC AND PROBABILISTIC MOVEMENT PRIMITIVES

We first briefly cover the fundamental aspects of DMPs. Then, we derive the analytical solution of the DMPs’ ordinary differential equation (ODE) to develop our new ProDMP representation. For convenience, we introduce our approach through a 1-DoF dynamical system and later extend it to a high-DoF system.

A. Solving DMPs’ Ordinary Differential Equation

For a single movement execution as a trajectory $\lambda = [y_i]_{i=0:T}$, [1], [8] model it as a second-order linear dynamical system with a non-linear forcing function $f$,

$$\tau^2 \ddot{y} + \alpha (\beta (y - \tau \dot{y}) + f(x)),$$

$$f(x) = \sum \varphi_i(x)w_i = x\varphi^T w,$$ (1)

where $y = y(t), \dot{y} = dy/dt, \ddot{y} = d^2y/dt^2$ represent the position, velocity, and acceleration of the system at time step $t$, respectively. Here, we use the original formulation of DMPs in [1] without any extensions. $\alpha$ and $\beta$ are spring-damper constants, $g$ is a goal attractor, and $\tau$ is a time constant which can be used to adapt the execution speed of the resulting trajectory. To this end, DMPs define the forcing function over an exponentially decaying phase variable $x(t) = e^{\alpha_x(t)/\tau}$, where $\varphi_i(x)$ represents the (unnormalized) basis functions and $w_i, i = 1 \ldots N$ are the corresponding weights. The trajectory of the motion $\lambda$ is obtained by integrating the dynamical system, i.e., applying numerical integration from starting time to the target time point. The dynamical system defined in (1) is a second-order linear non-homogeneous ODE with constant coefficients, whose closed-form solution can be derived analytically. We rewrite (1) and its homogeneous counterpart in standard form as

$$\text{DMPs’ ODE} : \dot{\dot{y}} + \frac{\alpha}{\tau^2}\dot{y} + \frac{\alpha \beta}{\tau^2}y = \frac{f(x)}{\tau^2} + \frac{\alpha \beta}{\tau^2}g \equiv F(x, g),$$ (2)

$$\text{Homo. ODE} : \dot{\dot{y}} + \frac{\alpha}{\tau^2}\dot{y} + \frac{\alpha \beta}{\tau^2}y = 0,$$ (3)

where $F$ denotes some function of $x$ and $g$. Using the method of variation of constants [24], the closed-form solution of the second-order ODE in (2), i.e., the trajectory position, is

$$y = c_1y_1 + c_2y_2 - y_1 \int \frac{y_2F}{Y} dt + y_2 \int \frac{y_1F}{Y} dt,$$ (4)

where $y_1, y_2$ are two linearly independent complementary functions of the homogeneous ODE given in (3). $c_1, c_2$ are two constants which are determined by the initial condition (IC) of the ODE, and $Y = y_1y_2 - y_1y_2$. Both integrals in (4) are indefinite. With appropriate values $\beta = \alpha/4$ [1], [8], the system
is critically damped, and the corresponding characteristic equation of the homogeneous ODE, i.e., \[
\Delta = (\alpha^2 - 4\alpha\beta)/\tau^2
\]
will be 0. Consequently, \(y_1, y_2\) will take the form
\[
y_1 = y_1(t) = \exp\left(-\frac{\alpha}{2\tau} t\right), \quad y_2 = y_2(t) = t \exp\left(-\frac{\alpha}{2\tau} t\right).
\]
(5)

Using this result, the term \(Y\) can also be simplified to \(Y = \exp(-\alpha t/\tau) \neq 0\). To get \(y\), we need to solve the two indefinite integrals in (4) as
\[
I_1(t) = \int \frac{y_2 F(x,y)}{Y} dt = \int t \exp\left(-\frac{\alpha}{2\tau} t\right) F(x,y) dt,
\]
\[
I_2(t) = \int \frac{y_1 F(x,y)}{Y} dt = \int \exp\left(-\frac{\alpha}{2\tau} t\right) F(x,y) dt.
\]
(6)

Applying the Fundamental Theorem of Calculus, i.e., \(\int h(t) dt = \int_{t_0}^{t} h(t) dt' + c\), \(c \in \mathbb{R}\), together with the definition of the forcing function \(f\) in (1) and \(F(x,y)\) in (2), \(I_1(t)\) can be expressed as
\[
I_1(t) = \frac{1}{\tau^2} \left[ \int_0^t t' \exp\left(\frac{\alpha}{2\tau} t'\right) x(t') \varphi_x^2 (\Delta) \Phi^T \varphi_2^T dt' \right. \\
\left. + \int_0^t t' \exp\left(\frac{\alpha}{2\tau} t'\right) \frac{\alpha}{4} y dt' \right] + c_3
\]
\[
= \frac{1}{\tau^2} \left[ \int_0^t t' \exp\left(\frac{\alpha}{2\tau} t'\right) x(t') \varphi_x^2 (\Delta) \Phi^T \varphi_2^T dt' \right] w \\
+ \left[ \left(\frac{\alpha}{2\tau} t - 1\right) \exp\left(\frac{\alpha}{2\tau} t\right) + 1 \right] g + c_3,
\]
(7)

where \(c_3\) is a constant fixed by the IC. From (7) to (8), we move the time-independent parameters \(w\) and \(g\) out of their corresponding integrals. Notice that the remaining part of the second integral has an analytical solution. The remaining part of the first integral, however, has no closed-form solution because the basis functions \(\varphi_x\) may be arbitrarily complex. Denoting these integrals as
\[
p_1(t) = \frac{1}{\tau^2} \int_0^t t' \exp\left(\frac{\alpha}{2\tau} t'\right) x(t') \varphi_x^2 (\Delta) \Phi^T \varphi_2^T dt',
\]
\[
q_1(t) = \left(\frac{\alpha}{2\tau} t - 1\right) \exp\left(\frac{\alpha}{2\tau} t\right) + 1,
\]
(9)

where \(p_1\) is a \(N\)-dim vector and \(q_1\) a scalar, we can express \(I_1(t) = p_1(t)^T w + q_1(t) g + c_3\). Following the same steps, we can obtain a similar solution for \(I_2\), i.e., \(I_2(t) = p_2(t)^T w + q_2(t) g + c_4\), where we present \(p_2(t)\) and \(q_2(t)\) in (10).
\[
p_2(t) = \frac{1}{\tau^2} \int_0^t t' \exp\left(\frac{\alpha}{2\tau} t'\right) x(t') \varphi_x^2 (\Delta) \Phi^T \varphi_2^T dt',
\]
\[
q_2(t) = \frac{\alpha}{2\tau} \left[ \exp\left(\frac{\alpha}{2\tau} t\right) - 1 \right]
\]
(10)

### B. DMPs’ Linear Basis Functions Representation.

Substituting the two integrals \(I_1\) and \(I_2\) in (4) by their derived form, the constants \(c_3\) and \(c_4\) can then be merged into \(c_1\) and \(c_2\), respectively. We can now express the position of DMPs in (4) as a summation of complementary functions \(y_1\) and \(y_2\), plus a linear basis function representation of the weights \(w\) and the goal attractor \(g\)
\[
y = c_1 y_1 + c_2 y_2 + [y_2 p_2 - y_1 p_1] y_2 q_2 - y_1 q_1 \left[ \begin{array}{c} w \\ g \end{array} \right]
\]
\[
\equiv c_1 y_1 + c_2 y_2 + \Phi^T w_g,
\]
(11)

where \(w_g\) is a \(N+1\)-dim vector containing \(w\) and \(g\). The resulting position basis functions for \(w\) and \(g\) are represented by \(\Phi(t)\), which can be solved numerically, cf. Fig. 1. The constants \(c_1\) and \(c_2\) are determined by solving a IC problem where we use the current position and velocity of the robot to inscribe where the trajectory should start.

### C. Solving the Initial Condition Problem

To compute the coefficients \(c_1\) and \(c_2\), we need to first obtain the velocity representation by computing the derivative of (4) w.r.t. time. The resulting equation reads
\[
\dot{y} = c_1 \dot{y}_1 + c_2 \dot{y}_2 - \dot{y}_1 \int \frac{y_2 F(x,y)}{Y} dt + \dot{y}_2 \int \frac{y_1 F(x,y)}{Y} dt.
\]
(12)

Note that (4) and (12) share a similar structure using the same constants \(c_1\) and \(c_2\), as well as the two indefinite integrals in (6). The only difference is that the two complementary functions \(y_1\) and \(y_2\) are replaced by their derivatives \(\dot{y}_1\) and \(\dot{y}_2\). By reusing the derivation of (6), (8), (9), (10), the trajectory velocity is ultimately represented by
\[
\dot{y} = c_1 \dot{y}_1 + c_2 \dot{y}_2 + [y_2 \dot{p}_2 - \dot{y}_1 p_1] \dot{y}_2 q_2 - \dot{y}_1 q_1 \left[ \begin{array}{c} w \\ g \end{array} \right]
\]
\[
\equiv c_1 \dot{y}_1 + c_2 \dot{y}_2 + \dot{\Phi}^T w_g,
\]
(13)

where \(\dot{\Phi}(t)\) represents the basis functions of the velocity. The position and velocity in (11), (13) share the same linear model structure, coefficients \(c_1, c_2\), and the parameters \(w_g\).

We need two initial conditions to solve \(c_1\) and \(c_2\). Theoretically, there are three options,
1) two position values at two time steps, i.e., \(y(t_{b_1}), y(t_{b_2})\), \(t_{b_1} \neq t_{b_2}\),
2) two velocity values at two time steps, i.e., \(\dot{y}(t_{b_1}), \dot{y}(t_{b_2})\), \(t_{b_1} \neq t_{b_2}\), and
3) one position value plus one velocity value \(y(t_{b_1}), \dot{y}(t_{b_2})\), where \(t_{b_1}\) and \(t_{b_2}\) are typically identical.

The third option allows us to naturally integrate the current position and velocity of a robot as initial conditions of a trajectory to be generated.

We denote the initial conditions as a position and velocity pair \(y(0), \dot{y}(0)\) at time step \(t_0\). The values of the complementary functions in (5) and their corresponding derivatives at \(t_0\) are denoted by \(y_1(t_0), y_2(t_0)\), \(\dot{y}_1(t_0), \dot{y}_2(t_0)\). The value of the position basis and velocity basis at \(t_0\) are denoted as \(\Phi_b, \Phi_v\). By substituting these terms into (11), (13), we can solve
\[
c_1 = \left[ \begin{array}{c} y_{1b} - y_{2b} y_1(t_0) - y_{1b} y_{2b} y_2(t_0) \\ y_{1b} - y_{2b} y_1(t_0) - y_{1b} y_{2b} y_2(t_0) \end{array} \right] + \left[ \begin{array}{c} y_{1b} y_2(t_0) - y_{1b} y_{2b} \Phi_v \Phi_v^T w_g \\ y_{1b} y_2(t_0) - y_{1b} y_{2b} \Phi_v \Phi_v^T w_g \end{array} \right],
\]
(14)
shows that the trajectory position $\dot{y}_D = \dot{y} \Psi_{\mu} + \dot{\Phi} + \Phi^T \omega_y$. (14)

where $\dot{y}_1 = \dot{y}_2 \Psi_{\mu} - \dot{y}_1 \Psi_{\mu}$, $\dot{y}_2 = \dot{y}_1 \Psi_{\mu} - \dot{y}_2 \Psi_{\mu}$, $\dot{y}_3 = \dot{y}_4 \Psi_{\mu} - \dot{y}_3 \Psi_{\mu}$, $\dot{y}_4 = \dot{y}_3 \Psi_{\mu} - \dot{y}_4 \Psi_{\mu}$.

(15) shows that the trajectory position $\dot{y}_i$ is fully determined by the initial position $\dot{y}_i$, and velocity $\dot{y}_i$ at the time step $t$, as well as the learnable weights $\omega_y$.

D. Probability Distribution of Multi DoF DMPs

The linear basis function representation of ProDMP takes the same form of ProMPs. Hence, our model can also be extended to multi-DoF systems $y = [y^1, \ldots, y^D]^T$, where $D$ denotes the system’s DoF. Similar to [2], we extend the basis functions $\Phi$, $\Psi$, as well as their initial values $\Phi_0$, $\Psi_0$ to block-diagonal matrices $\Phi$, $\Psi$, $\Phi_0$, $\Psi_0$, and concatenate each DoF’s initial conditions into a vector, e.g., $y_1^1 \ldots, y_1^D \rightarrow y_1 = [y_1^1, \ldots, y_1^D]^T$. Additionally, we note that the coefficient constants $c_1$ and $c_2$ of each DoF can be solved independently. As a consequence, we have the multi-DoF trajectory as $y = \xi_1 y_1 + \xi_2 y_2 + [\xi_3 \Psi_{\mu} + \xi_4 \Psi_{\mu} + \Psi]^T \omega_y$, (16)

Extending (16) from a single time step to the entire trajectory $\Lambda = [y_1]_t=0:T$, we can now express the trajectory distribution similarly to ProMPs. We consider a system where the current robot state $y_1, y_2, \ldots, y_n$ can be acquired precisely, which is universal in the most robotic systems. In this case, the trajectory’s variability is only caused by the variability of the weights $\omega_y$ plus the observation white noise $\epsilon$. Similar to ProMPs, assuming the weights $\omega_y$ follows a multivariate normal distribution $\omega_y \sim N(\mu_w, \Sigma_w)$, we can compute the trajectory distribution with full covariance over all time steps $0 : T$ and all DoF $1 : D$ as $p(\Lambda; \mu_w, \Sigma_w, y_0, \dot{y}_0) = N(\Lambda; \mu_\Lambda, \Sigma_\Lambda)$,

$\mu_\Lambda = \xi_1 y_1 + \xi_2 y_2 + H_{0:T} \mu_w$,

$\Sigma_\Lambda = H_{0:T} \Sigma_w H_{0:T} + \sigma_n^2 I$, (17)
IV. EMBED PRODMP IN A DEEP ARCHITECTURE USING BAYESIAN SET ENCODERS

We extend ProDMP with a deep NN architecture that allows for conditioning the trajectory distribution on a set of high-dimensional (time-stamped) sensory observations, such as images. Similar to recent MP architectures [13], [25], we treat such observations as set-inputs [26], [27], i.e., our architecture is invariant to the order of these observations.

Architecture: The architecture has four major parts: encoder, aggregator, decoder, and a ProDMP layer, cf. Fig. 4. The Encoders $E_p$, $E_s$, $D_p$, and $D_s$ compute for each observation $o_m \in \mathcal{O}$ a corresponding latent observation $r_m$ and an uncertainty $\sigma^2_{r_m}$, measuring how informative this observation is. The BA [16] is a parameter-free operator used to aggregate a set of latent observations $\{r_m\}$ and their uncertainties $\{\sigma^2_{r_m}\}$ into a latent state posterior $p(z|\mathcal{O})$ which is given by a factorized multivariate Gaussian distribution $\mathcal{N}(z|\mu_z, \sigma^2_z)$. The resulting aggregation is

$$\sigma^2_{z|O_{1:M}} = \left[ (\sigma^2_{z|0})^2 + \sum_{m=1}^{M} (\sigma^2_{r_m})^2 \right]^{\frac{1}{2}},$$

$$\mu_{z|O_{1:M}} = \mu_{z|0} + \sigma^2_z [\sum_{m=1}^{M} (r_m - \mu_{z|0})] \odot \sigma^2_{r_m},$$

(18)

where $\odot$, $\ominus$, and $\odot$ denote element-wise inversion, product, and division, respectively. The parameters of the latent variable’s prior distribution are $\mu_{z|0}$ and $\sigma^2_{z|0}$. Intuitively, BA emphasizes the learning of observation uncertainty $\sigma^2_{r_m}$ and allows to quantify the amount of information contained in an observation. A higher uncertainty will be translated into a little weight of the observation used in the aggregation procedure. Thus, BA can handle task ambiguity more efficiently in the aggregation result than the mean aggregation which recognizes each observation as equally important [16]. Hence, we adopt this approach for our architecture. The Decoders $D_p$, $D_s$ predict the mean $\mu_{wp}$ and $\mu_{ws}$ and the Cholesky decomposition $L_{wp}$ of the covariance $\Sigma_{wp}$ of the weights distribution. Similar to [16], we do not sample from the latent posterior $p(z|\mathcal{O})$ but use its mean $\mu_z$, and variance $\sigma^2_z$ to predict $\mu_{wp}$ and $L_{wp}$, respectively. Given the query time steps $\{t\}$ and the current robot position $y_b$ and velocity $\dot{y}_b$ as initial conditions, the ProDMP layer computes the parameters $\mu_A$ and $\Sigma_A$ of the trajectory distribution as a multivariate Gaussian distribution.

Loss Function: In this letter, we focus on Imitation Learning (IL) tasks and minimize the negative log-likelihood of the conditional trajectory distribution, i.e., $-\log \mathcal{N}(\cdot|\mu_A, \Sigma_A)$. Here, $\Lambda = \{y_t\}_{t=0...T}$ is the trajectory ground truth, and $\mu_A, \Sigma_A$ the mean and covariance of the predicted trajectory distribution conditioned on a set of observations $\mathcal{O}$. Yet, predicting a trajectory’s full covariance matrix $\Sigma_A$ demands high computational resources. The $TD \times TD$ covariance matrix has to be inverted in the loss computation. To keep the computation manageable, we never compute the distribution of the whole time sequence, but only compute the covariance of a pair of time steps, which limits the size of covariance matrices to $2D \times 2D$. We randomly select $J$ such time pairs $\{t_j, t'_j\}_{j=1,...,J}$, where $t, t' \in \{0, \ldots, T\}$, and predict the joint distribution on each of these pairs. As such random selection is executed in every training batch, we can still learn the correlation between time steps while keeping the cost of matrix inversion manageable. Given the IC $y_b$, $\dot{y}_b$, and a set of observations $\mathcal{O}$, the loss function is thus defined as the mean negative log-likelihood of $J$ random pairs of trajectory values $y(t, t')$, as

$$\mathcal{L}_\theta(\Lambda, y_b, \dot{y}_b, \mathcal{O}) = -\frac{1}{J} \sum_{j=1}^{J} \log \mathcal{N}(y(t_j, t'_j)|\mu_{(t, t')_j}, \Sigma_{(t, t')_j}),$$

(19)

where $\mu_{(t, t')_j}$ and $\Sigma_{(t, t')_j}$ denote the mean and covariance of the joint distribution at the $j$-th paired time points which are obtained from the ProDMP decoder depicted in Fig. 4. In Fig. 5, we illustrate that our loss function can capture the temporal correlation. Further, an ablation study that only computes the likelihood of single-time steps results in a degenerated estimate of the temporal correlation.

V. EXPERIMENTS

We highlight the advantages of our model and distinguish it from other methods through four experiments, which answer the following questions:

1) Does our model accelerate trajectory computation?

2) Does our model produce high-quality trajectory distributions and samples conditioning on high dimensional observations?

3) Does it support online replanning and predicting a smooth trajectory distribution from the current robot state?

4) Can we condition on several partial observations and obtain a trajectory distribution that leverages the aggregated information?

We compare our method with state-of-the-art CNMP [13] and NN-based DMPs models [9], [10], [11] on three digit-writing tasks using images as inputs, one simulated robot pushing task with complex physical interaction, and a real robot picking task with shifting object positions.
In Table I, we compare the computation time of generating a 2-DoF, 6-second long, 1000-Hz trajectory using the two pipelines shown in Fig. 3. We predict a 22-dim $w_i$ from a 3-layer fully connected network (10, 128, 22 neurons resp.). Our model is 200-4600 times faster than the NN-DMPs in different settings, which will translate into a speed-up of 10 times in further full learning experiments.

### A. Trajectory Computation Time Comparison

| Pipelines          | FP    | FP + IC | BP   | BP + IC |
|--------------------|-------|---------|------|---------|
| NN-DMPs            | 0.6057 s | 0.6145 s | 1.5261 s | 1.5737 s |
| ProDMPs            | 0.00013 s | 0.0027 s | 0.00105 s | 0.0039 s |
| Speed-up           | $\times$ 4659 | $\times$ 227 | $\times$ 1453 | $\times$ 403 |

### B. Learning Trajectory Distributions of the MNIST Digits

We use a synthetic-MNIST dataset [10] to showcase several properties of our method and advantages over the other approaches. The dataset contains 20,000 digit images in ten groups (0-9) and as prediction targets 3 seconds trajectories with 2 DoF each. In our settings, we choose to use 25 basis functions per DoF and predict a 52-dim multivariate Gaussian distribution over ProDMP parameters (25 weights and 1 goal for each DoF). A 2-dim vector for the starting point of the trajectory is also predicted. We analyse our experiments based on three subtasks and show the spatial writing trajectories, i.e., the x- and y-axis are spatial DoF.

**Comparison of Generative Models:** As first task, we consider the prediction of trajectories using different methods given one input image. For CNMP and ProDMP, we initially predict a distribution and then sample trajectories from it. As shown in Fig. 6, CNMP only model the mean and the isotropic variance per time step and thus fail to model correlations across time steps and dimensions. The NN-DMPs instead predicted a smooth trajectory, but do not learn any statistics over the trajectories and cannot generate samples. Our model, however, not only captures the correct shape of the trajectory but can also be used as a generative model to sample temporal and DoF consistent trajectories. To evaluate the speed of a full learning experiment, we replace the numerical integration of the NN-DMPs with our linear basis function model and keep the network architecture and MSE loss unchanged. The whole training time is reduced from 105 minutes to 10 minutes, which is approximately a speed-up by a factor of 10.

---

**Conditioning on Partial Observations:** For the second task, we show how our model leverages BA to deal with multiple partial observations. We repeatedly add synthetic noise to each original image such that we obtain three noisy images for each digit. The random noise masks occlude most of the image, leaving only a few original pixels visible. We use our model to progressively make trajectory distribution predictions conditioned on one, two, and three noisy images and sample from each resulting distribution. In Fig. 7, we can see our model aggregated the information contained in different observations and provided a refinement progressively (row by row). Without sufficient information, the trajectories appear random or misclassified but become refined once more information is added.

**Online Replanning:** Finally, we show how our method benefits from its dynamical system in online replanning once new information is provided. Different from the previous experiment where the noisy images are given before execution, in this task, we provide new observations to the model during the execution of the trajectory. The same noisy images as in the previous experiment of the target digit are provided at 0%, 25%, and 50% of the execution time. For illustration purposes, we trained the models only on images containing one type of digit or digits with similar spatial and temporal structure. As shown in Fig. 8, each additional observation increases the accuracy of the trajectory, while the new trajectories transition smoothly at the replanning points.

**C. Pushing and Replanning With Complex Contact**

We conduct a simulated robot experiment to prove that our approach can deal with rich physical interactions and online
conditioning on task variables. We use a Franka Panda robot in Mujoco [28] to push a box with its end-effector, cf. Fig. 9. The box has a square shape and is empty. The end-effector is equipped with a peg and starts in the box. Using a simple teleoperation interface, we control the end-effector’s \( x \) and \( y \) position with a mouse, which is then translated into the joint angles using inverse kinematics (IK). We collected 225 demos, each 3 seconds long, moving the box from a random initial state to a desired target location and orientation. The box is hard to control, especially the orientation, due to the non-linear interaction of its walls and corners with the peg. The learning procedure is as follows. At each iteration, we randomly choose a replanning time and use the recorded box position, orientation, as well as the actual robot end-effector’s position and velocity in the past 0.1 s as observations. We predict a desired end-effector trajectory distribution in the next 0.5 s using ProDMP with 25 basis functions per DoF and maximize the log-likelihood of CNMP are lacking temporal and DoF correlation, leading to an extremely low level of smoothness. The ablated case of our model cannot solve the task properly, indicating that replanning is crucial to adjust the movement in such a contact-rich scenario. The comparably high smoothness is achieved by the lack of replanning in the trajectory. The state-action-based behavioral cloning using an MSE loss cannot capture the temporal correlation between the box and the robot’s movement, and thus performs poorly. It is worth mentioning that using NN-based DMPs in the current task is equivalent to training our model with the MSE loss, which will lead to the same result but a faster speed.

### D. Object Picking With Dynamic Positional Shift

Finally, we learn an object-picking task in a 7-DoF joint space. A Franka Panda robot picks an object in its workspace (cf. Fig. 11(a)). During execution, the object shifts to a new position, and the robot must plan a new trajectory from its current state to the new object position without interruption. We collected 100 human demos, each 4 seconds long, using the same teleoperation interface as in Section V.C. In each demo, the human moves the robot end-effector to the object and adjusts the movement when the object shifts. The demonstrated joint trajectory is computed using IK. After each second, the actual object position and the robot state are given to ProDMP to replan.
a trajectory distribution starting from the current robot state. We evaluated the picking success rate of our model using the
1) mean prediction, and
2) trajectory samples in a simulated and a real setup.

The box position in the real setup is captured by a camera system using ArUco ROS [29]. Our model learns picking and replanning movement directly from the data and achieves a high success rate in simulation, as shown in Table III. Due to the camera delay in the real setup, the performance decreases slightly. Fig. 11(b) shows four predicted trajectory distributions in one sequence. Our model predicts the trajectory distribution with a high variance at the beginning and decreases the predicted variance once it observes the shift of the object.

VI. CONCLUSION

We presented ProDMP, a unified framework fusing dynamic and probabilistic movement primitives. ProDMP recovered a linear basis-function representation for the trajectories by solving the ODE of the dynamical system. This way, we can easily represent trajectory distributions that adhere to initial conditions defined by the current robot position and velocity as well as generate smooth trajectories when replanning. Further, we built a neural aggregation model for non-linear iterative conditioning and found a solution to learn full trajectory covariances with fewer resources. Our deep embedded ProDMP achieved smoothness, goal convergence, trajectory correlation modeling, non-linear trajectory conditioning, and online replanning in one model. For future work, we will extend our approach to reinforcement learning and consider force profiles that need to be applied. We expect our model can process data with multi-sensor modalities and thus achieve sensor fusion in the latent space. If the data has strong temporal order and causality, using a recurrent structure or a transformer may increase the performance. For robotic experiments, using raw camera images as input will make the tasks more challenging.

REFERENCES

[1] S. Schaal, “Dynamic movement primitives—a framework for motor control in humans and humanoid robotics,” in Adaptive motion of animals and machines. Berlin, Germany: Springer, 2006.
[2] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” in Proc. 26th Int. Neural Inf. Process. Syst., 2013, pp. 2616–2624.
[3] G. J. Maeda, G. Neumann, M. Ewerton, R. Lioutikov, O. Kroemer, and J. Peters, “Probabilistic movement primitives for coordination of multiple human–robot collaborative tasks,” Auton. Robots, vol. 41, pp. 593–612, 2017.
[4] Y. Zhou, J. Gao, and T. Asfour, “Learning via-point movement primitives with inter-and extrapolation capabilities,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2019, pp. 4301–4308.

[5] O. Celik, D. Zhou, G. Li, P. Becker, and G. Neumann, “Specializing versatile skill libraries using local mixture of experts,” in Proc. Conf. Robot Learn., 2022, pp. 1423–1433.
[6] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, “Phase estimation for fast action recognition and trajectory generation in human–robot collaboration,” Int. J. Robot. Res., vol. 36, pp. 1579–1594, 2017.
[7] F. Otto, O. Celik, H. Zhou, H. Ziesche, V. A. Ngo, and G. Neumann, “Deep black-box reinforcement learning with movement primitives,” in Proc. 6th Ann. Conf. Robot Learn., 2022.
[8] A. J. Ijspeert, J. Nakaniishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” Neural Comput., vol. 25, pp. 328–373, 2013.
[9] R. Pahić, A. Gams, A. Ude, and J. Morimoto, “Deep encoder-decoder networks for mapping raw images to dynamic movement primitives,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 5863–5868.
[10] R. Pahić, B. Ridge, A. Gams, J. Morimoto, and A. Ude, “Training of deep neural networks for the generation of dynamic movement primitives,” Neural Netw., vol. 127, pp. 121–131, 2020.
[11] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, “Neural dynamic policies for end-to-end sensorimotor learning,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 5058–5069.
[12] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, “Using probabilistic movement primitives in robotics,” Auton. Robots, vol. 42, pp. 529–551, 2018.
[13] M. Y. Seker, M. Imre, J. H. Piater, and E. Ugur, “Conditional neural movement primitives,” in Proc. Robot.: Sci. Syst., 2019, vol. 10.
[14] F. Brandner, J. Peters, G. Neumann, and R. Akour, “Learning replanning policies with direct policy search,” IEEE Robot. Automat. Lett., vol. 4, no. 2, pp. 2196–2203, Apr. 2019.
[15] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, “Dynamic movement primitives in robotics: A tutorial survey,” 2021, arXiv:2102.03861.
[16] M. Volpp, Flürenbrock, L. Grossberger, C. Daniel, and G. Neumann, “Bayesian context aggregation for neural processes,” in Proc. Int. Conf. Learn. Representations, 2021.
[17] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, “Statistical dynamical systems for skills acquisition in humanoids,” in Proc. 12th IEEE-RAS Int. Conf. Humanoid Robots, 2012, pp. 323–329.
[18] C. Yang, C. Chen, W. He, R. Cui, and Z. Li, “Robot learning system based on adaptive neural control and dynamic movement primitives,” IEEE Trans. neural Netw. Learn. Syst., vol. 30, no. 3, pp. 777–787, Mar. 2019.
[19] F. Meier and S. Schaal, “A probabilistic representation for dynamic movement primitives,” 2016, arXiv:1612.05932.
[20] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, “Interaction primitives for human-robot cooperation tasks,” in Proc. IEEE Int. Conf. Robot. Automat., 2014, pp. 2831–2837.
[21] J. Urain, M. Ginesi, D. Tateo, and J. Peters, “ImitationFlow: Learning deep stable stochastic dynamic systems by normalizing flows,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 5231–5237.
[22] S. M. Khansari-Zadeh and A. Billard, “Learning control Lyapunov functions to ensure stability of dynamical-system based robot reaching motions,” Robot. Auton. Syst., vol. 62, pp. 752–765, 2014.
[23] M. Garnefo et al., “Conditional neural processes,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 1704–1713.
[24] W. E. Boyce, R. C. DiPrima, and D. B. Meade, Elementary Differential Equations and Boundary Value Problems. Hoboken, NJ, USA: Wiley, 2021.
[25] M. Akbulut, E. Oztop, M. Y. Seker, X. Hh, A. Tekden, and E. Ugur, “ACNMP: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing,” in Proc. Conf. Robot Learn., 2021, pp. 1896–1907.
[26] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and M. I. Jordan, “Deep sets,” in Proc. Adv. Neural Inf. Process. Syst., 2017, vol. 30.
[27] M. Garnefo et al., “Neural processes,” 2018, arXiv:1807.01622.
[28] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2012, pp. 5026–5033.
[29] R. Salinas and B. Magyar, “Aruco: A minimal library for augmented reality applications based on opencv,” Universidad de Córdoba, vol. 386, 2015.