# From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control

Julius Jankowski[1,2], Mattia Racca[1], and Sylvain Calinon[1,2]

***Abstract*—In the field of Learning from Demonstration (LfD), movement primitives learned from full trajectories provide mechanisms to generalize a demonstrated skill to unseen situations. Key position demonstrations, requiring the user to provide only a sequence of via-points rather than a complete trajectory, have been shown to be an appealing alternative. In this letter, we investigate the synergy between learning adaptive movement primitives and key position demonstrations. We exploit a linear optimal control formulation to (1) recover the timing information of the skill missing from key position demonstrations, and to (2) infer low-effort movements on-the-fly. We evaluate the performance of the proposed approach in a user study where 16 novice users taught a 7-DoF robot manipulator, showing improved learning efficiency and trajectory smoothness. We further showcase the effectiveness of the approach for tasks that require precise demonstrations and on-the-fly movement adaptation.**

***Index Terms*—Learning from Demonstration; Imitation Learning; Machine Learning for Robot Control**

## I. Introduction

FOR robots to be widely adopted across applications and environments, it is critical that a wide range of users can program robots with as little effort as possible. State-of-the-art Learning from Demonstration (LfD) techniques enable robots to learn from demonstrations, *i.e.*, exemplary behaviours of a target task. Demonstrations often consist of the time-series of relevant variables, *e.g.*, the robot's end-effector position or its joint configuration. Movement primitive methods learn from such *full trajectory* demonstrations, providing mechanisms to adapt the robot's motion to situations unseen during teaching.

Typically, methods based on movement primitives provide little prior knowledge to the learning procedure. As a consequence, the demonstrations need to teach the robot not only the task at hand but also desirable characteristics of the robot's motion, such as the smoothness of the trajectory. Even with intuitive interfaces like kinesthetic teaching, it can be challenging for novice users to provide full trajectory demonstrations that successfully complete the task and, simultaneously, convey the desired characteristics [1]–[3].

Manuscript received: September, 9, 2021; Revised November, 9, 2021; Accepted January, 5, 2022. This paper was recommended for publication by Editor D. Popa upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the CODIMAN project (Swiss National Science Foundation), and by the ROSALIS project (Swiss National Science Foundation).

[1]Julius Jankowski, Mattia Racca and Sylvain Calinon are with the Robot Learning and Interaction Group, Idiap Research Institute, Martigny, Switzerland julius.jankowski@idiap.ch

[2]Julius Jankowski and Sylvain Calinon are with the EPFL, Lausanne, Switzerland.

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

As an alternative to full trajectories, the literature has explored the use of *key position* demonstrations, requiring the user to demonstrate only a sparse set of consecutive poses. This alternative interface alleviates some issues of kinesthetic teaching, chiefly the need of teachers to provide smooth trajectories [4], letting the user focus on the task of programming [3], [5] or teaching a successful task execution [6]. Furthermore, the teacher is not required to demonstrate with the same pace as the desired robot executions, allowing for *e.g.*, more accurate placements of key positions. We expect this aspect to facilitate in particular the kinesthetic teaching of high-precision robot tasks such as peg-in-hole.

We propose an approach that learns adaptive movement primitives from key position demonstrations, as illustrated in Fig. 1. Given an ordered set of key positions, our method fills the gaps between them by recovering the temporal information missing from the demonstrations (*i.e.*, when to reach the key positions) and generating a smooth-by-design distribution of trajectories (*i.e.*, how to reach the key positions). Similarly to other motion primitive approaches, the learned trajectory distribution is encoded as a stochastic, linear combination of basis functions. To learn the timing for the robot to reach the key positions, a cascaded time-optimal control problem is solved for each key position demonstration and a common set of basis functions that capture a conservative timing is extracted. Furthermore, by including information about the task context and leveraging variability in the demonstrations, our method can adapt the generated robot trajectories to unseen scenarios. Paired with the robot controller presented in [7], the generated trajectories adapt on-the-fly based on the current state of both the robot and the environment.

To evaluate the proposed LfD pipeline, we conducted a user study, where 16 participants provided either full trajectory or key position demonstrations to teach a 7 DoF manipulator a pick-and-place task. Results show that the proposed key position method can generate trajectories as successfully (in terms of task completion) as a Probabilistic Movement Primitive (ProMP) [8] learned from full trajectories, while being smoother. Furthermore, we showcase how our method can be used to solve a peg-in-hole task and a pushing task purely from key position demonstrations.

## II. Related Work

### A. Key Position LfD

Demonstrating a motion, either by showing the full trajectory or by recording a sequence of via-points, is a commonly

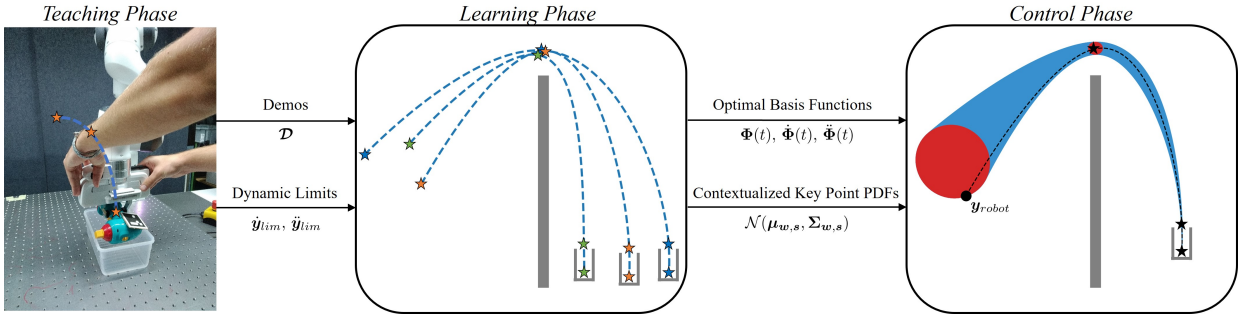


Fig. 1. *Teaching Phase*: The user provides contextualized demonstrations as a set of ordered key positions (asterisks) and dynamic limits. *Learning Phase*: Time-optimal trajectories (blue-dashed curve) are computed in order to construct an optimal basis for the stochastic key position space. *Control Phase*: By computing the distribution of the key positions in the given context (red ellipses), a probability distribution of optimal trajectories (blue area) is computed and used to efficiently infer an optimal reference trajectory (black-dashed curve) based on the current robot state $\boldsymbol{y}_{\text{robot}}$.

adopted interface for programming robots [9]–[11]. Both modalities provide the user with an intuitive programming interface, while differing in terms of advantages and disadvantages. Full trajectory demonstrations, especially when paired with gravity compensated robots and kinesthetic teaching, are an intuitive interface for novice users. While allowing expert users to demonstrate the dynamics of motions, demonstrating high quality full trajectories can be challenging, especially as the number of degrees of freedoms raises [12]. Programming robots with key positions can produce more readable and editable robot programs, while removing the dynamic component from the demonstration and letting the user focus solely on the definition of key positions [3], [4]. It has been shown however that users find more mentally taxing to specify key positions and to form mental models of collision avoidance, compared to full trajectories [4].

Via-Point Movement Primitives (VMPs) [13] are an hybrid approach that learns from full trajectories that can be later offset to pass through desired via-points. While allowing for skill extrapolation, VMPs still require full trajectory demonstrations to learn, therefore inheriting the aforementioned limitations. Another hybrid learning approach that accepts as input both full trajectory and key position demonstrations is proposed in [4], [6], converting full trajectory demonstrations into a key position representation in order to merge them.

While showing successful skill reproduction capabilities, these approaches can not learn solely from key position demonstrations a skill with variations. Furthermore, they lack the capability to generalize the skill to unseen situations (*e.g.*, novel starting robot configurations or target object locations), and to adapt on-the-fly as the environment changes (lack of adaptability).

### B. Adaptive Movement Primitives

While both modalities (and their hybrids, like *e.g.*, automatic extraction of via-points from demonstrations [3], [14]) have been thoroughly adopted in robot programming interfaces, the literature on learning adaptive movement primitives has been tightly linked to the concept of full trajectory demonstrations.

To represent a single trajectory or a distribution of trajectories in a compact form, Dynamical Movement Primitives (DMPs) [15] and ProMPs [8] rely on a predefined set of basis functions (typically, radial basis functions with uniform spacing and constant bandwidth). Alternatively, approaches based on (Bayesian) Gaussian Mixture Regression (GMR) [16] first encode the trajectories as a joint distribution of space and time/phase variables, and then use the conditional property of Gaussians for regression, providing an approach to automatically estimate the number and placement of the basis functions. As shown in [17], probabilistic trajectory learning and adaptation can also be formulated as an optimal control problem, by considering a standard Linear Quadratic Regulator (LQR) with full precision matrices and a virtual system in the form of a simple or double integrator.

In [7], a controller for ProMPs has been proposed that enables online adaptation to dynamically changing task contexts. In the following, we show how our approach learns, starting from key position demonstrations, a probabilistic movement primitive with optimally chosen basis functions. This allows us to reuse the controller from [7] and obtain adaptable movement primitives from key position demonstrations. In Section V, we therefore compare the proposed approach against a baseline of ProMPs learned from full trajectories.

## III. Learning Phase

During the learning phase, the user input is processed into a distribution of optimal trajectories. The user provides a set of $K$ demonstrations $\boldsymbol{\mathcal{D}} = \{\boldsymbol{\mathcal{D}}_k\}_{k=1}^{K}$, with $N$ key positions per demonstration, *i.e.*, $\boldsymbol{\mathcal{D}}_k = \{\{{}^k\boldsymbol{y}_n\}_{n=1}^{N}, \boldsymbol{s}_k\}$. The proposed approach can be applied in the robot joint space as well as in the the robot's end-effector space. In the latter case, the presented approach applies to the end-effector position only and needs to be complemented by an additional orientation controller (see Section IV-B). $\boldsymbol{s}_k$ reflects the state of the environment during the $k$-th demonstration. While the $K$ demonstrations show different solutions of one task, we assume that the number of key positions is constant among all demonstrations and that the order of the key positions is given by the user. In contrast to learning from full trajectories, there is no information about the timing of the demonstrated task (except for the order in which key positions are given). We recover this information by means of optimal control by exploiting the velocity and acceleration limits we want to impose. These limits are given (*e.g.*, by the user) in the form of element-wise intervals, *e.g.*, $\dot{y}^i \in [-\dot{y}^i_{\text{lim}}, \dot{y}^i_{\text{lim}}]$ and $\ddot{y}^i \in [-\ddot{y}^i_{\text{lim}}, \ddot{y}^i_{\text{lim}}]$ for the $i$-th DoF.

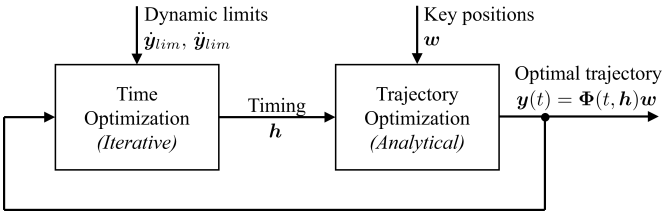


Fig. 2. Within the learning phase, we solve a cascaded optimization problem for each key position demonstration. We exploit linear relations in the resulting optimal trajectory by extracting basis functions $\boldsymbol{\Phi}(t, \boldsymbol{h})$ for a demonstrated skill.

The learning phase consists of (1) constructing optimal basis functions (Section III-A) and (2) learning a distribution of optimal trajectories by learning a probability density function of the basis function weights (Section III-B) plus correlations with the task context (Section III-C). In our approach, the basis function weights explicitly correspond to the $N$ key positions and the start and end velocities, *i.e.*, $\boldsymbol{w} = (\boldsymbol{y}_1^\top, \boldsymbol{y}_2^\top, \ldots, \boldsymbol{y}_N^\top, \dot{\boldsymbol{y}}_1^\top, \dot{\boldsymbol{y}}_T^\top)^\top$.

### A. Optimal Basis Functions

In order to construct optimal basis functions, we formulate a cascaded time-optimal control problem. The block diagram in Fig. 2 gives an overview of the cascaded optimization scheme. We exploit the fact that the result of this optimization problem is linear in $\boldsymbol{w}$.

*1) Trajectory Optimization:* The inner optimal control problem considers a linear second-order system that is constrained to pass through key positions $\boldsymbol{y}_n$ at a given time $t_n$ with minimal effort, *i.e.*,

$$\begin{aligned} \boldsymbol{y}^*(t) = \arg\min_{\boldsymbol{y}(t)} \quad & \int_0^T \ddot{\boldsymbol{y}}(t)^\top \ddot{\boldsymbol{y}}(t) dt, \\ \text{s.t.} \quad & \boldsymbol{y}(t_n) = \boldsymbol{y}_n, \quad n = 1, ..., N, \\ & \dot{\boldsymbol{y}}(0) = \dot{\boldsymbol{y}}_1, \quad \dot{\boldsymbol{y}}(T) = \dot{\boldsymbol{y}}_N. \end{aligned} \tag{1}$$

An instance of the optimal control problem in (1) can be represented by $\boldsymbol{w}$ and a corresponding timing parameter $\boldsymbol{h} \in \mathbb{R}^{N-1}$ that encodes the time passed between two consecutive key positions, *i.e.*, $h_n = t_{n+1} - t_n$. Consequently, the total duration of the motion is given by $T = \sum_{n=1}^{N-1} h_n$. In [18], it has been shown that the solution of the linear optimal control problem in (1) is a polynomial spline in time. We find that the optimal trajectory can be expressed as a linear function of $\boldsymbol{w}$ (see Appendix for details), *i.e.*,

$$\boldsymbol{y}^*(t) = \boldsymbol{\Phi}(t)\boldsymbol{w}. \tag{2}$$

The elements of the time-dependent matrix $\boldsymbol{\Phi}(t)$ can be interpreted as optimal basis functions that, weighted by the elements of $\boldsymbol{w}$, retrieve an optimal trajectory for the key positions encoded in $\boldsymbol{w}$. Fig. 3 shows a set of basis functions for a task instance with three key positions with a general timing parameter $\boldsymbol{h}$. Each colored signal represents a basis function that corresponds either to one of the key positions or to one of the boundary velocities.

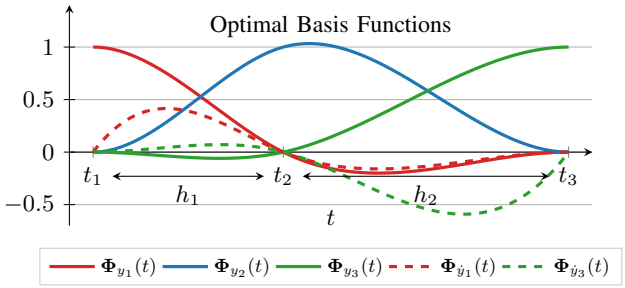


Fig. 3. Basis functions constructed from the optimal control problem in (1) for a single DoF and $N = 3$ key positions.

*2) Time Optimization:* Within the outer optimization problem, we optimize the timing parameter $\boldsymbol{h}$. Since the inner optimization problem uniquely translates a timing parameter $\boldsymbol{h}$ into an optimal basis $\boldsymbol{\Phi}(t)$, the timing parameter can be seen as a direct input to the optimal basis functions (*i.e.*, $\boldsymbol{\Phi}(t, \boldsymbol{h}) = \boldsymbol{\Phi}(t)$). Thus, we formulate the outer optimization problem as

$$\begin{aligned} \boldsymbol{h}^* = \arg\min_{\boldsymbol{h}} \; & T, \\ \text{s.t.} \quad & h_n > 0, \\ & \dot{y}^i(t) = \dot{\boldsymbol{\Phi}}^i(t, \boldsymbol{h})\boldsymbol{w} \in [-\dot{y}^i_{\text{lim}}, \dot{y}^i_{\text{lim}}], \\ & \ddot{y}^i(t) = \ddot{\boldsymbol{\Phi}}^i(t, \boldsymbol{h})\boldsymbol{w} \in [-\ddot{y}^i_{\text{lim}}, \ddot{y}^i_{\text{lim}}], \end{aligned} \tag{3}$$

with $\dot{y}^i(t)$, $\ddot{y}^i(t)$ being the velocity and acceleration of the $i$-th DoF, such that $\dot{\boldsymbol{\Phi}}^i(t, \boldsymbol{h})$ and $\ddot{\boldsymbol{\Phi}}^i(t, \boldsymbol{h})$ are the $i$-th rows of $\dot{\boldsymbol{\Phi}}(t)$ and $\ddot{\boldsymbol{\Phi}}(t)$, respectively. We find a timing parameter $\boldsymbol{h}^*$ that locally minimizes the movement duration $T$ while satisfying the constraints through the sequential least squares quadratic programming algorithm [19]. For each demonstration of key positions $\boldsymbol{\mathcal{D}}_k$ (setting $\dot{\boldsymbol{y}}_1 = \dot{\boldsymbol{y}}_N = \boldsymbol{0}$), we compute the optimal timing $\boldsymbol{h}_k^*$ and select a conservative timing parameter $\bar{\boldsymbol{h}}$ by selecting each element (*i.e.*, the individual duration between consecutive key positions) as the maximum that appeared among all demonstrations. This conservative timing parameter $\bar{\boldsymbol{h}}$ is used to compute offline the coefficients of a common set of optimal basis functions $\boldsymbol{\Phi}(t, \bar{\boldsymbol{h}})$, simply referred to as $\boldsymbol{\Phi}(t)$ in the following sections.

### B. Probability Distribution of Optimal Trajectories

The result of the previously described cascaded optimization is a common basis $\boldsymbol{\Phi}(t)$ for all demonstrated key positions. Note that the mathematical structure of an optimal trajectory in (2) given a set of key positions resembles the structure of a ProMP. Unlike the hand-crafted basis functions of ProMPs, the basis functions of our approach are implicitly learned from the demonstrations by means of optimal control. By demonstrating ordered key positions for a given task context, the user provides direct samples of an underlying joint distribution $p(\boldsymbol{w}, \boldsymbol{s})$ modelling the task through correlations between key positions and task contexts. We assume that the provided samples $\boldsymbol{\mathcal{D}}$ can be modelled by a Gaussian Mixture Model (GMM) with a finite number of components, *i.e.*,

$$\boldsymbol{w}, \boldsymbol{s} | \boldsymbol{\mathcal{D}} \sim \sum_c \mathcal{N}({}^c\boldsymbol{\mu}_{\boldsymbol{w},\boldsymbol{s}}, {}^c\boldsymbol{\Sigma}_{\boldsymbol{w},\boldsymbol{s}}). \tag{4}$$

Each component of the GMM is translated into a controller that is activated if the system state and the task context can be explained through the corresponding demonstrations of the component (see [7] for details). Thus without loss of generality, we consider a single Gaussian distribution in the remainder for the sake of readability.

The optimal, contextualized robot position at a given time $t$ can be inferred as

$$\boldsymbol{y}(t)|\boldsymbol{s},\boldsymbol{\mathcal{D}} \sim \mathcal{N}\Big(\boldsymbol{\Phi}(t)\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Phi}(t)\boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{s}}\boldsymbol{\Phi}^\top(t)\Big), \quad (5)$$

with $\boldsymbol{w}|\boldsymbol{s},\boldsymbol{\mathcal{D}} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{s}})$ being the basis function weight distribution conditioned on the state of the environment $\boldsymbol{s}$. The blue area in the right-hand block in Fig. 1 depicts a time-dependent Gaussian distribution resulting from (5).

### C. Learning in Multiple Frames

The task model in (4) captures piecewise linear relations between the key positions and the state of the environment. However, orientations of objects in the environment can impose nonlinear relations that would require multiple components in (4). In [17], poses of objects in the environment are specifically handled by learning the movement statistics in a coordinate system that is attached to the corresponding object. The same can be applied to our approach, resulting in multiple probability distributions of optimal trajectories, with $p_{\mathcal{M}}(\boldsymbol{w}|\boldsymbol{s},\boldsymbol{\mathcal{D}}) = \mathcal{N}(\boldsymbol{\mu}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Sigma}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}})$ being the distribution learned in a coordinate system $\mathcal{M}$. Since the timing of the task is equal in all coordinate systems, the optimal basis functions $\boldsymbol{\Phi}(t)$ remain the same in all coordinate systems. The online fusion of multiple distributions can thus be done in the basis function weights by computing a product of Gaussians, after transforming all basis function weight distributions into a world-fixed coordinate system $\mathcal{W}$, as

$$\begin{aligned}\boldsymbol{w}|\boldsymbol{s},\boldsymbol{\mathcal{D}} &\sim \mathcal{N}(\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}) \\ &= \prod_{\mathcal{M}} \mathcal{N}(\bar{\boldsymbol{R}}_{\mathcal{M}}\boldsymbol{\mu}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}} + \bar{\boldsymbol{t}}_{\mathcal{M}}, \bar{\boldsymbol{R}}_{\mathcal{M}}\boldsymbol{\Sigma}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}}\bar{\boldsymbol{R}}^\top_{\mathcal{M}}),\end{aligned} \quad (6)$$

where $\bar{\boldsymbol{R}}_{\mathcal{M}}$ and $\bar{\boldsymbol{t}}_{\mathcal{M}}$ transform a $\boldsymbol{w}$ that is expressed in frame $\mathcal{M}$ into the world-fixed coordinate system $\mathcal{W}$.

## IV. Control Phase

During the control phase, a controller generates motor commands allowing the robot to reproduce the learned behavior by feeding back the robot state as well as the state of the environment. In Section IV-A, we recall the concept of probabilistic adaptive control and apply it to the previously derived movement primitive. Section IV-B provides a controller design for the orientation to complete the control of the end-effector pose. The control commands derived in different spaces are composed in the robot's joint space in realtime as an information fusion problem (product of Gaussians).

### A. Probabilistic Adaptive Control

In [7], we designed the controller as a force/torque control action, which has the benefit that the time-varying feedback gains are interpretable as mechanical compliance. However, for composing multiple control policies as in [20], acceleration control actions are better suited. Thus, we define a trajectory tracking controller, *i.e.*,

$$\boldsymbol{a} = -\boldsymbol{K}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}) - \boldsymbol{D}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{\Phi}}\boldsymbol{w}) + \ddot{\boldsymbol{\Phi}}\boldsymbol{w}, \quad (7)$$

that forces the system to track an optimal trajectory that is given by $\boldsymbol{y}_{des}(t) = \boldsymbol{\Phi}(t)\boldsymbol{w}$. Here, the matrices $\boldsymbol{K}$ and $\boldsymbol{D}$ are design parameters. As the basis function weights $\boldsymbol{w}$ are Gaussian-distributed, the mean of the acceleration control action is inferred as

$$\boldsymbol{\mu}_{\boldsymbol{a}|\boldsymbol{x},\boldsymbol{s}} = -\tilde{\boldsymbol{K}}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}) - \tilde{\boldsymbol{D}}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{\Phi}}\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}) + \ddot{\boldsymbol{\Phi}}\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}, \quad (8)$$

with $\boldsymbol{x}$ being the robot state composed of position and velocity, and

$$\begin{aligned}\tilde{\boldsymbol{K}} &= \boldsymbol{K} - (\boldsymbol{K}\boldsymbol{\Phi} + \boldsymbol{D}\dot{\boldsymbol{\Phi}} + \ddot{\boldsymbol{\Phi}})\boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{x},\boldsymbol{s}}\boldsymbol{\Phi}^\top\boldsymbol{\Sigma}^{-1}_{\boldsymbol{y}}, \\ \tilde{\boldsymbol{D}} &= \boldsymbol{D} - (\boldsymbol{K}\boldsymbol{\Phi} + \boldsymbol{D}\dot{\boldsymbol{\Phi}} + \ddot{\boldsymbol{\Phi}})\boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{x},\boldsymbol{s}}\dot{\boldsymbol{\Phi}}^\top\boldsymbol{\Sigma}^{-1}_{\dot{\boldsymbol{y}}}.\end{aligned} \quad (9)$$

The matrix $\boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{x},\boldsymbol{s}}$ is the covariance of the basis function weights conditioned on the current robot state and the state of the environment. The matrices $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ and $\boldsymbol{\Sigma}_{\dot{\boldsymbol{y}}}$ represent the expected tracking error, *e.g.*, due to modelling errors of the system dynamics.

### B. Orientation Control

For full pose control of the robot end-effector, we propose to combine the probabilistic adaptive controller learned from the key positions with a reactive orientation controller learned from the corresponding key orientations. We adopt the idea of probabilistic adaptive control for orientations by defining an angular velocity control command as $\boldsymbol{\omega}_d = \boldsymbol{K}_{\text{ori}}(\boldsymbol{\Delta\theta} - \boldsymbol{\Delta\theta}_d)$, with $\boldsymbol{\Delta\theta} = \text{Log}_q(\mu_q)$ the logarithmic map at the robot's end-effector quaternion $q$ and $\boldsymbol{\Delta\theta}_d \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}_d})$. In this formulation, the parameters $\mu_q$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_d}$ are directly learned from the orientations provided during the demonstrations within the coordinate system of interest (*e.g.*, the coordinate system of an object to grasp). Analogue to the tracking error matrices $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ and $\boldsymbol{\Sigma}_{\dot{\boldsymbol{y}}}$, we introduce a control error matrix for the orientation controller by modelling $\boldsymbol{\Delta\theta} \sim \mathcal{N}(\boldsymbol{\Delta\theta}_d, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. Consequently, the conditional mean of the desired angular velocity simplifies to

$$\boldsymbol{\mu}_{\boldsymbol{\omega}_d|\boldsymbol{\Delta\theta}} = \boldsymbol{K}_{\text{ori}}\left(\boldsymbol{\Sigma}^{-1}_{\boldsymbol{\theta}} + \boldsymbol{\Sigma}^{-1}_{\boldsymbol{\theta}_d}\right)^{-1}\boldsymbol{\Sigma}^{-1}_{\boldsymbol{\theta}_d}\boldsymbol{\Delta\theta}. \quad (10)$$

Finally, the angular acceleration control command is defined as $\boldsymbol{a}_{\boldsymbol{\theta}} = -\boldsymbol{D}_{\text{ori}}(\boldsymbol{\omega} - \bar{\boldsymbol{\mu}}_{\boldsymbol{\omega}_d|\boldsymbol{\Delta\theta}})$, with $\boldsymbol{\omega}$ being the angular velocity of the robot end-effector and $\bar{\boldsymbol{\mu}}_{\boldsymbol{\omega}_d|\boldsymbol{\Delta\theta}}$ being the clipped result of (10) in order to limit the angular velocity on-the-fly.

## V. Experiments

We here present the results of a user study comparing the effectiveness of teaching through key positions (Method **KP**) against full trajectories (Method **FT**), alongside other experiments showcasing the effectiveness of the proposed approach.

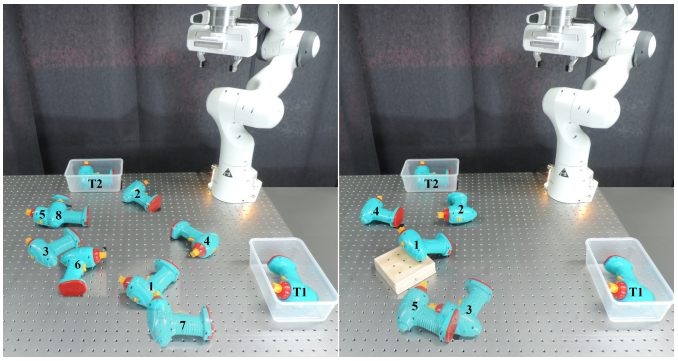


Fig. 4. Experimental setup of the user study: drill pick-up (1-8, left figure) and drop locations (T1 and T2) used for the demonstration collection, and test locations used in the evaluation phase (1-5, right figure).

### A. User Study

We conducted a study where 16 novice robot users (mean age 28.8, SD 3.2) provided demonstrations to solve a pick-and-place task with a 7 DoF FRANKA EMIKA Panda arm. This study was approved by Idiap Research Institute's Data and Research Ethics Committee.

*1) Experimental Setup:* The task consisted of picking a toy drill (from the YCB dataset [21]) and place it inside one of two boxes, as shown in Fig. 4. The participants provided 8 demonstrations by displacing the gravity-compensated robot arm. The number of demonstrations was selected as a trade-off between data collection and a sensible duration of the study (mean duration of 25 minutes). For each demonstration, a drill pose was randomly selected from a pool of predefined demonstration scenarios, along with a corresponding robot's starting configuration. Variability in the selection of scenarios was systematically enforced to prevent participants from providing uninformative demonstrations, *e.g.*, multiple demonstrations with the same drill pose.

*2) Conditions and Protocol:* Participants first filled a brief questionnaire consisting of three 7-point Likert scale statements, aimed at assessing their robotics expertise. An average score of 1.95 was observed (Cronbach's $\alpha = 0.9$), indicating novice users. After a brief familiarization phase with the gravity-compensated robot, each participant provided 8 demonstrations for the aforementioned pick-and-place task, for one of two methods (between-subjects study design): the proposed via-points demonstrations (Method **KP**, presented in Section III), and full trajectory demonstrations (Method **FT**, acting as baseline). Each method was therefore used by 8 participants.

For Method **KP**, the participants were instructed to perform the task by using 5 key positions, *i.e.*, 3 for the pick action (including the starting pose) and 2 for the place action. Participants displaced the robot in its workspace (kinesthetic teaching) and added key positions by uttering a verbal command (*e.g.*, *"Insert here!"*). For Method **FT**, the participants solely needed to verbally specify the beginning and the end of their demonstrations. For Method **FT**, the number of radial basis functions was set to 24, evenly split between the pick and the place actions. Since the basis functions of Method **FT** are equally distanced in time, a larger number of them compared to Method **KP** is required, in order to capture local details (*e.g.*, the grasping pose). For both methods, the task context (*i.e.*, the drill pose) is captured by learning in multiple frames (as presented in Section III-C), thus no explicit task context variable $\boldsymbol{s}$ was used for the user study. The trajectory of the robot end-effector pose in the world frame (at the base of the robot) and in the object frame (located on the drill) was recorded, although Method **KP** used only the provided key positions for the learning. To facilitate the learning for Method **FT**, we pre-processed the recorded end-effector position trajectory by automatically finding the start- and end time step of each demonstration and cutting off the idle parts of the trajectory. The pre-processed data is then used to learn a ProMP representation of the task for Method **FT**, following the algorithm in [22].

The average computation time of the learning phase with 8 demonstrations for one participant is 4.6 seconds for Method **KP** and 4.2 seconds for Method **FT**. For learning the key position timing for Method **KP**, we empirically set the velocity limit to $\dot{y}^i_{lim} = 0.15\ \frac{\mathrm{m}}{\mathrm{s}}$ and the acceleration limit to $\ddot{y}^i_{lim} = 2\ \frac{\mathrm{m}}{\mathrm{s}^2}$ (equal for all axes). Since both learning approaches result in the mathematical form given in (4), we apply the inference in (6) to both task representations (with slight modifications for the ProMP) and consequently use the control design in Section IV for both approaches with the same hyper-parameters. A demonstration was considered failed if either the pick or the place actions were unsuccessful (*e.g.*, the drill slipped from the robot gripper or was placed outside of the designated boxes). Failed demonstrations were excluded from the learning and no corrective demonstrations were collected.

### B. Evaluation

For each participant, we learned task-parameterized trajectory distributions, following the method in Section III for the participants providing demonstrations consisting of key positions (Method **KP**), and by learning a ProMP for Method **FT**. Three feedback controllers as in (8) were learned for each participant, with as input the first 4, 6, and all successful demonstrations to evaluate how this impacts the quality of the skill reproduction. Each controller was tested on 5 novel static scenarios, different from the ones used during the demonstration collection (see Fig. 4). These static drill poses are prerecorded and provided to the controllers, avoiding computer vision inaccuracies to impact the results.We additionally tested the controllers in a mock-up handover task, where a second robot arm[1] pushes the drill on the table towards the controlled robot, equipped with a camera at the wrist to track the drill's pose. This scenario requires the trajectory to be adapted on-the-fly, therefore testing the online adaptation capabilities of the proposed approach. For each reproduction on a novel static scenario, we computed the acceleration effort $e$, defined as

$$e = \int_0^T \ddot{\boldsymbol{y}}(t)^\top \ddot{\boldsymbol{y}}(t) dt, \tag{11}$$

[1]The second robot performs two fixed (although unknown to the first robot) pushing actions with a short intermediate pause, ensuring a consistent experimental scenario.

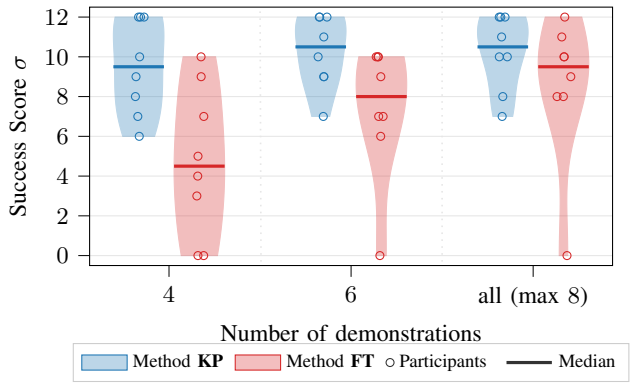


Fig. 5. Distribution of cumulative success score $\sigma$, separated by method and by number of demonstrations utilized for training.

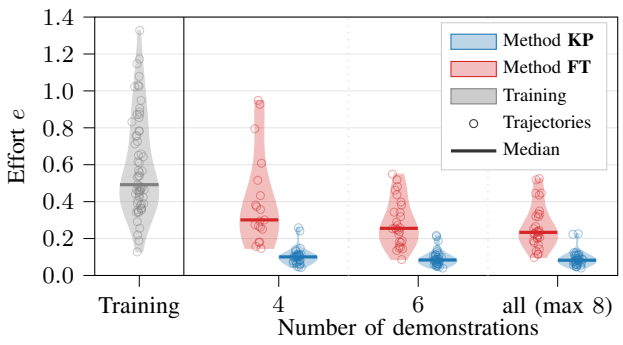


Fig. 6. Distribution of acceleration effort $e$ of trajectory reproductions, separated by method and by number of demonstrations utilized for training. The leftmost violin instead shows $e$ of the collected demonstrations.

where $\ddot{\boldsymbol{y}}(t)$ is the recorded acceleration of the robot end-effector, therefore measuring the smoothness of the reproduced trajectory.

We further adopted a scoring system to operationalize the success of the task: 1 point for a successful pick (0 otherwise), and 1 point for a successful place (0 otherwise), for a maximum of 2 points per reproduction. The pick action was considered successful if the drill did not slip from the closed gripper during the transport. The place action was considered successful if the drill landed in either of the two boxes. We refer to the cumulative success obtained by a participant with *e.g.*, 4 demonstrations and Method **KP** as $\sigma^4_{\text{KP}}$, ranging from 0 (failure on all 6 novel scenarios) to $2 \times 6 = 12$ (complete success).

### C. Results

We compare the participants' effectiveness of teaching with Method **KP** and Method **FT** and with different numbers of demonstrations, in terms of success of the task reproductions and of quality of the reproduced trajectory. Informed by the literature on teaching through key positions [3], [4], we expected

1) the two methods to achieve similar success scores (with the score improving with a larger number of demonstrations), and
2) Method **KP** to produce smoother trajectories than Method **FT**, thanks to the optimization procedure described in Section III-A.

We furthermore expected both methods to improve, in terms of trajectory smoothness, over their input, *i.e.*, the participants' demonstrations.

Fig. 5 compares the distribution of cumulative score $\sigma$ for each teaching method. We observe a statistically significant score difference between methods when trained with 4 and 6 demonstrations (Mann–Whitney U test[2], $p<.01^{**}$), with a median score of 9.5 for Method **KP** vs 4.5 for Method **FT** when trained with 4 demonstrations. This difference can be explained by the fact that Method **KP** needs to learn in a lower-dimensional space than Method **FT**; a relevant difference when learning from few demonstrations. As expected, $\sigma$ improves for both methods with more than 6 demonstrations, with negligible differences between methods.

Fig. 6 compares the distribution of acceleration effort $e$ for each teaching method and for the participants' demonstrations. Unsurprisingly, we see how the reproduced trajectories of both methods require lower acceleration effort $e$ compared to the participants' kinesthetic demonstrations. Method **KP** produces however trajectories with significantly lower $e$ compared to Method **FT** (Mann–Whitney U test, $p<.01^{**}$ for any number of demonstrations used in the training). This is a direct consequence of the optimization process presented in Section III-A.

The differences between Method **KP** and Method **FT** are further accentuated when looking at the hand-over task (see the accompanying video for qualitative results). With 4 demonstrations, Method **KP** succeeded at the task for 5 participants out of 8. The number of successful executions goes to 6 when trained with all the available demonstrations. In contrast, no successful execution was possible with Method **FT**, when trained with 4 and 6 demonstrations. Even when using all available demonstrations, Method **FT** obtains successful executions only for 2 of the 8 participants.

We observe that the main reason for failing at grasping the drill is that the robot, with the camera mounted on its wrist, looses track of the moving object while approaching it due to abrupt movements not directed towards the drill. The presented results, including the higher effort required for static scenarios (see Fig. 6), indicate that Method **FT** is more prone to produce those abrupt movements compared to Method **KP**.

As also observed in previous work [4], participants are almost twice as fast when providing full trajectory demonstrations (median 18 s) compared to key position demonstrations (median 28 s). However, the success score achieved by Method **KP** trained with 4 demonstrations ($\sigma^4_{\text{KP, median}} = 9.5$) is comparable with the one of Method **FT** with twice the demonstrations ($\sigma^8_{\text{FT, median}} = 9.5$). We argue that, in a real scenario, operations other than the demonstration collection, such as setting the environment or the robot arm's initial configuration, would dominate the process, making the aforementioned time difference negligible.

[2]Data was tested for normality, always rejecting the null hypothesis (Shapiro-Wilk test, $p<.01^{**}$). We therefore used non-parametric tests.

### D. Showcase Tasks

We evaluate the effectiveness of our approach with expert key position demonstrations for two tasks. See the accompanying video for results.

*1) Peg-in-Hole:* The robot is taught to insert a peg into a tight-tolerance hole at a fixed location while avoiding an obstacle. Since the environment is constant during the task, no context variable is learned for this experiment. The robot reproduces the task from unseen initial positions while being physically perturbed.

*2) Plate Pushing:* The robot is taught to keep a plate at the center of a table by applying local pushes, informed by feeding back the current position of the plate. The task context $\boldsymbol{s}$ is chosen to be the first two coordinates (horizontal plane) of the plate in a frame of reference located at the robot's base. The robot reproduces the task while the plate position is dynamically changed.

## VI. Conclusion

In this letter, we presented an approach to learning adaptive movement primitives from key position demonstrations. To recover the time information of these demonstrations, lost due to the nature of teaching with key positions, our approach minimizes the movement duration while keeping the velocity and acceleration in bounds with a cascaded time-optimal control formulation, resulting in optimal basis functions. In a user study with novice users, we evaluated the effectiveness of our key position approach as a teaching interface against a full trajectory baseline method. We showed that learning meaningful basis functions through linear optimal control reduces the number of kinesthetic demonstrations required for a successful skill reproduction in unseen scenarios. Furthermore, we showed how the learned basis functions enforce low-effort movements even when adapting to unseen situations on-the-fly.

A limitation of the proposed approach is that the timing of the skill is computed offline. As such, adapting the robot movement online does not preserve time-optimality w.r.t. (3) (optimality w.r.t. (1) is inherently preserved). We plan to investigate an extension of the proposed approach to orientation manifolds in order to learn pose movement primitives. Furthermore, we plan to investigate the combination of the learned probabilistic adaptive controller and additional reactive controllers, such as controllers for realtime collision avoidance exploiting the learned variations of the skill.

## References


[1] A. Weiss, J. Igelsboeck, S. Calinon, A. G. Billard, and M. Tscheligi, "Teaching a humanoid: A user study on learning by demonstration with HOAP-3," in *Proc. IEEE Intl Symp. on Robot and Human Interactive Communication (Ro-Man)*, (Toyama, Japan), pp. 147–152, September 2009.

[2] S. Wrede, C. Emmerich, R. Grünberg, A. Nordmann, A. Swadzba, and J. Steil, "A user study on kinesthetic teaching of redundant robots in task and configuration space," *J. Hum.-Robot Interact.*, vol. 2, no. 1, pp. 56–81, 2013.

[3] G. Ajaykumar, M. Stiber, and C.-M. Huang, "Designing user-centric programming aids for kinesthetic teaching of collaborative robots," *Robotics and Autonomous Systems*, p. 103845, 2021.

[4] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pp. 391–398, 2012.

[5] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pp. 453–462, IEEE, 2017.

[6] B. Akgun, M. Cakmak, K. Jiang, and A. L. Thomaz, "Keyframe-based learning from demonstration," *International Journal of Social Robotics*, vol. 4, no. 4, pp. 343–355, 2012.

[7] J. Jankowski, H. Girgin, and S. Calinon, "Probabilistic adaptive control for robust behavior imitation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1997–2004, 2021.

[8] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, 2018.

[9] T. Lozano-Perez, "Robot programming," *Proceedings of the IEEE*, vol. 71, no. 7, pp. 821–841, 1983.

[10] A. G. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), pp. 1371–1394, Secaucus, NJ, USA: Springer, 2008.

[11] S. Calinon, "Learning from demonstration (programming by demonstration)," in *Encyclopedia of Robotics* (M. H. Ang, O. Khatib, and B. Siciliano, eds.), Springer, 2019.

[12] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.

[13] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter- and extrapolation capabilities," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pp. 4301–4308, 2019.

[14] F. Steinmetz, V. Nitsch, and F. Stulp, "Intuitive task-level programming by demonstration through semantic skill recognition," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3742–3749, 2019.

[15] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1547–1554, 2002.

[16] S. Calinon, "Mixture models for the analysis, edition, and synthesis of continuous time series," in *Mixture Models and Applications* (N. Bouguila and W. Fan, eds.), pp. 39–57, Springer, 2019.

[17] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, (Hong Kong, China), pp. 3339–3344, May-June 2014.

[18] Z. Zhang, J. Tomlinson, and C. Martin, "Splines and linear control theory," *Acta Math. Appl*, vol. 49, pp. 1–34, 1997.

[19] D. Kraft, *A software package for sequential quadratic programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht, Wiss. Berichtswesen d. DFVLR, 1988.

[20] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv:1801.02854*, 2018.

[21] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[22] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Transactions on Robotics (T-Ro)*, no. 2, pp. 366–379, 2020.


## APPENDIX

### A. Computation of Optimal Basis Functions

In the following, we derive the computation of the optimal basis functions for one DoF.

In [18], it has been shown that the control

$$u_n(t) = \boldsymbol{b}^\top e^{-\boldsymbol{A}^\top (t-t_n)} \boldsymbol{M}(h_n)(e^{-\boldsymbol{A}h_n}\boldsymbol{x}_{n+1} - \boldsymbol{x}_n), \quad (12)$$

with $h_n{=}t_{n+1}{-}t_n$, minimizes the functional $J(u){=}\int_{t_n}^{t_{n+1}} u^2(\tau)d\tau$ among all controls that move the linear system $\dot{\boldsymbol{x}}{=}\boldsymbol{A}\boldsymbol{x}{+}\boldsymbol{b}u$ from $\boldsymbol{x}(t_n){=}\boldsymbol{x}_n$ to $\boldsymbol{x}(t_{n+1}){=}\boldsymbol{x}_{n+1}$. In this paper, we aim at minimizing the acceleration (*cf.* (1)), thus the linear system simplifies to a double-integrator

with $\boldsymbol{x}=(y,\dot{y})^\top$ and $u=\ddot{y}$. The optimal acceleration in (12) becomes a linear function in time

$$\ddot{y}_n(t) = [-(t-t_n),\ 1]\,\boldsymbol{M}(h_n)\begin{bmatrix} y_{n+1}-y_n-h_n\dot{y}_{n+1} \\ \dot{y}_{n+1}-\dot{y}_n \end{bmatrix}, \tag{13}$$

with

$$\boldsymbol{M}(h_n) = \left(\int_0^{h_n} e^{-\boldsymbol{A}\tau}\boldsymbol{b}\boldsymbol{b}^\top e^{-\boldsymbol{A}^\top\tau}d\tau\right)^{-1} = \tfrac{2}{h_n^3}\begin{bmatrix} 6 & 3h_n \\ 3h_n & 2h_n^2 \end{bmatrix}.$$

The optimal acceleration in (13) can be used to construct a spline that moves the system from $\boldsymbol{x}(0)=\boldsymbol{x}_1$, passing through $y(t_n)=y_n,\ n=2,\dots,N-1$, to $\boldsymbol{x}(T)=\boldsymbol{x}_N$ in an optimal way. Note that the $N-2$ velocities $\dot{y}(t_n)$ at intermediate key positions are unknown. Thus, we introduce $\boldsymbol{w}=(y_1,y_2,\dots,y_N,\dot{y}_1,\dot{y}_N)^\top$ as known parameters and $\boldsymbol{v}=(\dot{y}_1,\dot{y}_2,\dots,\dot{y}_N)^\top$ as unknown key velocities (note that $\dot{y}_1$ and $\dot{y}_N$ are known, but part of $\boldsymbol{v}$ for consistency of (14)). Formulating (13) as a function of $\boldsymbol{w}$ and $\boldsymbol{v}$ yields

$$\ddot{y}_n(t) = [-(t-t_n),\ 1]\,\boldsymbol{M}(h_n)\left({}^{\boldsymbol{w}}\boldsymbol{L}_n\boldsymbol{w}+{}^{\boldsymbol{v}}\boldsymbol{L}_n\boldsymbol{v}\right), \tag{14}$$

with ${}^{\boldsymbol{w}}\boldsymbol{L}_n\in\mathbb{R}^{2\times N+2}$ satisfying ${}^{\boldsymbol{w}}\boldsymbol{L}_n\boldsymbol{w}=[y_{n+1}-y_n,\ 0]^\top$, and ${}^{\boldsymbol{v}}\boldsymbol{L}_n\in\mathbb{R}^{2\times N}$ satisfying ${}^{\boldsymbol{v}}\boldsymbol{L}_n\boldsymbol{v}=[-h_n\dot{y}_{n+1},\ \dot{y}_{n+1}-\dot{y}_n]^\top$.

A unique optimal acceleration spline is obtained by imposing continuity, *i.e.*, $\ddot{y}_n(t_{n+1})=\ddot{y}_{n+1}(t_{n+1})$. This constraint forms a linear system of equations, *i.e.*, $\boldsymbol{P}_{\boldsymbol{v}}\boldsymbol{v}=\boldsymbol{P}_{\boldsymbol{w}}\boldsymbol{w}$, with $\boldsymbol{P}_{\boldsymbol{v}}\in\mathbb{R}^{N\times N}$ and $\boldsymbol{P}_{\boldsymbol{w}}\in\mathbb{R}^{N\times N+2}$ being defined by $N-2$ continuity constraints at intermediate key positions and 2 equality constraints for start and end velocity. Note that by solving for $\boldsymbol{v}=\boldsymbol{P}_{\boldsymbol{v}}^{-1}\boldsymbol{P}_{\boldsymbol{w}}\boldsymbol{w}$, we obtain optimal key velocities. Inserting the optimal key velocities back into (14), we obtain the optimal acceleration for segment $n$ as a linear function of the known parameters $\boldsymbol{w}$, *i.e.*,

$$\ddot{y}_n(t) = [-(t-t_n),\ 1]\,\boldsymbol{\Omega}_n\boldsymbol{w}, \tag{15}$$

with $\boldsymbol{\Omega}_n=\boldsymbol{M}(h_n)\left({}^{\boldsymbol{w}}\boldsymbol{L}_n+{}^{\boldsymbol{v}}\boldsymbol{L}_n\boldsymbol{P}_{\boldsymbol{v}}^{-1}\boldsymbol{P}_{\boldsymbol{w}}\right)$. The optimal acceleration at time $t\in[0,T]$ is then given by $\ddot{y}(t)=\ddot{\boldsymbol{\phi}}(t)\boldsymbol{w}$, with

$$\ddot{\boldsymbol{\phi}}(t) = [-(t-t_n),\ 1]\,\boldsymbol{\Omega}_n, \quad t_n < t < t_{n+1}. \tag{16}$$

Consequently, the optimal velocity and position splines are given through integration of $\ddot{y}(t)$, *i.e.*, $\dot{y}(t)=\dot{\boldsymbol{\phi}}(t)\boldsymbol{w}$ and $y(t)=\boldsymbol{\phi}(t)\boldsymbol{w}$ respectively, with

$$\begin{aligned}\dot{\boldsymbol{\phi}}(t) &= \left[-\frac{1}{2}(t-t_n)^2,\ t-t_n\right]\boldsymbol{\Omega}_n + {}^{\boldsymbol{v}}\boldsymbol{l}_n\boldsymbol{P}_{\boldsymbol{v}}^{-1}\boldsymbol{P}_{\boldsymbol{w}} \\ \boldsymbol{\phi}(t) &= \left[-\frac{1}{6}(t-t_n)^3,\ \frac{1}{2}(t-t_n)^2\right]\boldsymbol{\Omega}_n + \\ &\quad (t-t_n){}^{\boldsymbol{v}}\boldsymbol{l}_n\boldsymbol{P}_{\boldsymbol{v}}^{-1}\boldsymbol{P}_{\boldsymbol{w}} + {}^{\boldsymbol{w}}\boldsymbol{l}_n,\end{aligned} \tag{17}$$

with $n$ being defined by $t_n<t<t_{n+1}$, ${}^{\boldsymbol{v}}\boldsymbol{l}_n\in\mathbb{R}^N$ satisfying ${}^{\boldsymbol{v}}\boldsymbol{l}_n\boldsymbol{v}=\dot{y}_n$ and ${}^{\boldsymbol{w}}\boldsymbol{l}_n\in\mathbb{R}^{N+2}$ satisfying ${}^{\boldsymbol{w}}\boldsymbol{l}_n\boldsymbol{w}=y_n$. Note that for a given timing parameter $\boldsymbol{h}$, the time-independent coefficients (*e.g.*, $\boldsymbol{\Omega}_n$, $\boldsymbol{P}_{\boldsymbol{v}}^{-1}\boldsymbol{P}_{\boldsymbol{w}}$) in (17) can be precomputed once in order to reduce online computation cost.

For systems with $d$ DoF, such that $\boldsymbol{w}=(\boldsymbol{y}_1^\top,\dots,\boldsymbol{y}_N^\top,\dot{\boldsymbol{y}}_1^\top,\dot{\boldsymbol{y}}_N^\top)^\top$, with $\boldsymbol{y}_n\in\mathbb{R}^d$, the optimal position trajectory is given by $\boldsymbol{y}(t)=\boldsymbol{\Phi}(t)\boldsymbol{w}$, where the basis function matrix is given through a Kronecker product, *i.e.*, $\boldsymbol{\Phi}(t)=\boldsymbol{\phi}(t)\otimes\boldsymbol{I}_d$, with $\boldsymbol{I}_d\in\mathbb{R}^{d\times d}$ being the identity matrix.