eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

A Non-parametric Skill Representation with Soft Null Space Projectors for Fast Generalization

João Silvério[1] and Yanlong Huang[2]

Abstract— Over the last two decades, the robotics community witnessed the emergence of various motion representations that have been used extensively, particularly in behavorial cloning, to compactly encode and generalize skills. Among these, probabilistic approaches have earned a relevant place, owing to their encoding of variations, correlations and adaptability to new task conditions. Modulating such primitives, however, is often cumbersome due to the need for parameter re-optimization which frequently entails computationally costly operations. In this paper we derive a non-parametric movement primitive formulation that contains a null space projector. We show that such formulation allows for fast and efficient motion generation and adaptation with computational complexity $\mathcal{O}(n^2)$ without involving matrix inversions, whose complexity is $\mathcal{O}(n^3)$. This is achieved by using the null space to track secondary targets, with a precision determined by the training dataset. Using a 2D example associated with time input we show that our non-parametric solution compares favourably with a state-of-the-art parametric approach. For demonstrated skills with high-dimensional inputs we show that it permits on-the-fly adaptation as well.

I. INTRODUCTION

Generalization is a fundamental challenge in machine learning, which in robot skill learning translates into modifying a behavior when new conditions arise. Since their inception with dynamic movement primitives (DMP) [1], skill representations in robotics have targeted generalization. In DMPs this is done by design, by representing motion dynamics with a second-order linear system that ensures convergence to a pre-specified, alterable end-goal. Alternative approaches using mixture models and hidden Markov models [2] are also endowed with strong generalization capabilities particularly under the framework of task-parameterized movement models [3], [4]. Despite their popularity, these approaches lack analytical solutions to the adaptation problem, especially when via-points are not located at the start or end of the movement. In these situations, re-optimization of model parameters is required which can be prohibitively costly, e.g., reinforcement learning was used to optimize the parameters of DMP [5] and GMM [6]. In [7] DMPs were combined in sequence to address the via-point issue, demanding several DMPs for a single skill, as opposed to traditional solutions requiring only one DMP.

A second line of approaches leverages probability theory to provide increased adaptation capabilities. Paraschos et al. introduced probabilistic movement primitives (ProMP) [8].

ProMP assumes that demonstrations of a task are approximated by a parametric trajectory amounting to a weighted sum of basis functions and uses Gaussian conditioning to compute optimal weights to pass through new, desired via-points. ProMP has been extended to handle obstacle avoidance [9], [10], where in [9] a signed distance between a robot arm and an obstacle was used similarly to CHOMP [11], and in [10] a random exploration and update scheme, initialized by ProMP weight distributions, was provided similarly to STOMP [12]. In [13] constrained ProMP was studied by formulating external constraints into a nonlinear optimization problem. In [14], we introduced a non-parametric, kernelized movement primitives (KMP) formulation which uses kernel functions to mitigate the need to carefully define basis functions, a well-known issue in ProMP and related techniques relying on fixed basis functions. This is the case especially if the trajectory is not driven by time but by a multi-dimensional input, such as the position or posture of a human in the robot workspace, where the number of required basis functions grows quickly with the number of input variables [15]. We argue that formulations that allow to seamlessly encode dependencies on human motions, and not only time, have an important role to play in the near future as human-robot interaction gains traction [16]–[19].

Despite providing an elegant solution to adapt a (potentially multi-dimensional-input) trajectory to pass through regions that were not previously shown to the robot, the computational cost of dealing with such adaptations by KMP is non-negligible. This is because it relies on a kernel matrix that, depending on factors like the task and the features being encoded (e.g. robot end-effector poses, joint angles, velocities), may be costly to invert. It should be noted that this is an intrinsic limitation of many non-parametric methods, including Gaussian processes [20].

In this paper we address the fast adaptation problem by deriving a variant of KMP that contains a null space projector. We start from the original problem formulation, described in Section II, and modify the optimization objective such that a null space projector appears naturally (Section III-A). Null space projectors play an important role in robot control [21]–[25] and we re-use some of those core ideas here in a different domain. Subsequently we show that the projector can be kernelized too, providing an elegant and computationally efficient solution to probabilistic trajectory adaptation (Section III-B), including in problems with multi-dimensional inputs. We also discuss some of its properties (Section IV), particularly the fact that idempotence of the projector is not guaranteed and depends on the training

[1] German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchner Str. 20, 82234 Weßling, Germany. joao.silverio@dlr.de
[2] School of Computing, University of Leeds, Leeds LS29JT, UK. y.l.huang@leeds.ac.uk

data (thus we refer to it as a *soft* null space projector). While on the surface this might appear to be a limitation, it allows for covariance-weighted exploration trivially. Finally we compare our approach to two baselines in 2D and 3D examples (Sections V–VI).

II. KERNELIZED MOVEMENT PRIMITIVES

In the KMP framework we are interested in estimating a model that predicts the value of an output variable $\boldsymbol{\xi} \in \mathbb{R}^D$ given observations of an input $\boldsymbol{s} \in \mathbb{R}^I$ from a set of M datapoints $\{\boldsymbol{s}_m, \boldsymbol{\xi}_m\}_{m=1}^M$ typically, but not necessarily, collected from human demonstrations. Examples of $\boldsymbol{s}$ include time or human hand positions, while $\boldsymbol{\xi}$ is often the robot end-effector pose and its derivatives [14]. Similarly to ProMP, KMP assumes a parametric approximation of a trajectory by a weighted sum of basis functions, i.e.,

$$\boldsymbol{\xi}(\boldsymbol{s}) = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{s}) & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{\phi}(\boldsymbol{s}) \end{bmatrix}^\top \boldsymbol{w} = \boldsymbol{\Phi}(\boldsymbol{s})^\top \boldsymbol{w}, \text{ where } \boldsymbol{w} \in \mathbb{R}^{BD}$$

are weights and $\boldsymbol{\phi} \in \mathbb{R}^B$ is a vector of basis functions whose values depend on $\boldsymbol{s}$. Moreover, $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$. ProMP finds $\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w$ using maximum likelihood estimation (MLE) with samples of $\boldsymbol{w}$ obtained from demonstrations. KMP departs from ProMP by further assuming that a *reference trajectory distribution* $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^N$ is available to model $\mathcal{P}(\boldsymbol{\xi}|\boldsymbol{s}_n)$, where $\boldsymbol{s}_{n=1,\dots,N}$ are N given inputs. It then finds $\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w$ by minimizing the objective $J(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) = \sum_{n=1}^N \mathrm{D}_{\mathrm{KL}}\Big(\mathcal{N}(\boldsymbol{\Phi}(\boldsymbol{s}_n)^\top \boldsymbol{\mu}_w, \boldsymbol{\Phi}(\boldsymbol{s}_n)^\top \boldsymbol{\Sigma}_w \boldsymbol{\Phi}(\boldsymbol{s}_n)) || \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\Big)$. In this paper we focus on finding the optimal values of $\boldsymbol{\mu}_w^*$, which is done by solving the *regularized weighted least squares* problem (see [14] for the detailed derivation):

$$\boldsymbol{\mu}_w^* = \underset{\boldsymbol{\mu}_w}{\operatorname{argmin}} \sum_{n=1}^N \Big(\boldsymbol{\Phi}(\boldsymbol{s}_n)^\top \boldsymbol{\mu}_w - \boldsymbol{\mu}_n\Big)^\top \boldsymbol{\Sigma}_n^{-1} \Big(\boldsymbol{\Phi}(\boldsymbol{s}_n)^\top \boldsymbol{\mu}_w - \boldsymbol{\mu}_n\Big)$$
$$+ \lambda \boldsymbol{\mu}_w^\top \boldsymbol{\mu}_w, \quad (1)$$

where $\lambda > 0$ is a scalar. Differentiating the objective in (1) and equating to zero results in

$$\boldsymbol{\mu}_w^* = \underbrace{\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1}}_{(\boldsymbol{\Phi}^\top)^\dagger} \boldsymbol{\mu}, \quad (2)$$

where we have

$$\boldsymbol{\Phi} = [\boldsymbol{\Phi}(\boldsymbol{s}_1) \dots \boldsymbol{\Phi}(\boldsymbol{s}_N)], \quad \boldsymbol{\mu} = [\boldsymbol{\mu}_1^\top \dots \boldsymbol{\mu}_N^\top]^\top,$$
$$\boldsymbol{\Sigma} = \operatorname{blockdiag}(\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_N). \quad (3)$$

The underscored term in Eq. (2) $(\boldsymbol{\Phi}^\top)^\dagger$ can be interpreted as a regularized right pseudo-inverse of $\boldsymbol{\Phi}^\top$ with regularization term $\lambda \boldsymbol{\Sigma}$.

For a test input $\boldsymbol{s}^*$, the expectation of $\boldsymbol{\xi}(\boldsymbol{s}^*)$ under the assumption of a parametric trajectory is given by $\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{\Phi}(\boldsymbol{s}^*)^\top \boldsymbol{\mu}_w^*$. By replacing (2) and applying the *kernel trick* [15], we obtain

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{\Phi}(\boldsymbol{s}^*)^\top \boldsymbol{\mu}_w^* = \boldsymbol{\Phi}(\boldsymbol{s}^*)^\top \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\mu}$$
$$(4)$$
$$= \boldsymbol{k}^* \left(\boldsymbol{K} + \lambda \boldsymbol{\Sigma}\right)^{-1} \boldsymbol{\mu}, \quad (5)$$

where $\boldsymbol{k}^* = [\boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}_1), \dots, \boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}_N)]$,
$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{s}_1, \boldsymbol{s}_1) & \dots & \boldsymbol{k}(\boldsymbol{s}_1, \boldsymbol{s}_N) \\ \vdots & \ddots & \vdots \\ \boldsymbol{k}(\boldsymbol{s}_N, \boldsymbol{s}_1) & \dots & \boldsymbol{k}(\boldsymbol{s}_N, \boldsymbol{s}_N) \end{bmatrix}, \quad \boldsymbol{k}(\boldsymbol{s}_i, \boldsymbol{s}_j) = k(\boldsymbol{s}_i, \boldsymbol{s}_j)\boldsymbol{I}$$
and $k(\boldsymbol{s}_i, \boldsymbol{s}_j) = \boldsymbol{\phi}(\boldsymbol{s}_i)^\top \boldsymbol{\phi}(\boldsymbol{s}_j)$ is a kernel function from hereon assumed to be the squared-exponential $k(\boldsymbol{s}_i, \boldsymbol{s}_j) = \exp(-\frac{1}{l^2}||\boldsymbol{s}_i - \boldsymbol{s}_j||^2)$.

From (1), (5) it follows that if, for a certain $\boldsymbol{\mu}_n$, the covariance $\boldsymbol{\Sigma}_n$ is small, the expectation at $\boldsymbol{s}_n$ will be close to $\boldsymbol{\mu}_n$. This provides a principled way for trajectory modulation. Indeed, if, for a new input $\bar{\boldsymbol{s}}$, one wants to ensure that the expected trajectory passes through a desired $\bar{\boldsymbol{\mu}}$ it suffices to manually add the pair $\{\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\}$ to the reference distribution provided that $\bar{\boldsymbol{\Sigma}}$ is small enough. While this solves the adaptation problem trivially, it comes at the cost of having to invert the term $\boldsymbol{K} + \lambda \boldsymbol{\Sigma}$ every time a new point is added. This hinders the applicability of KMP in scenarios that require a fast evaluation of variations of the demonstrated trajectories. We propose to alleviate this issue by formulating trajectory modulation as acting on the null space of the movement primitive.

III. A NULL SPACE FORMULATION OF KMP

A. KMP with null space as the solution to a least squares problem

We take inspiration from the robotics literature on null space projectors [22]–[24] and extend the original problem (1) with an additional cost term to keep the solution close to a desired one denoted by $\hat{\boldsymbol{w}}$, i.e.,

$$\boldsymbol{\mu}_w^* = \underset{\boldsymbol{\mu}_w}{\operatorname{argmin}} \left(\boldsymbol{\Phi}^\top \boldsymbol{\mu}_w - \boldsymbol{\mu}\right)^\top \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\Phi}^\top \boldsymbol{\mu}_w - \boldsymbol{\mu}\right) + \alpha \boldsymbol{\mu}_w^\top \boldsymbol{\mu}_w$$
$$\underbrace{+ \beta \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}}\right)^\top \left(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}}\right)}_{J(\boldsymbol{\mu}_w)}. \quad (6)$$

The solution can be obtained similarly to (1) by solving $\frac{\partial J(\boldsymbol{\mu}_w)}{\partial \boldsymbol{\mu}_w} = 0$, leading to

$$\boldsymbol{\mu}_w^* = \left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + (\alpha + \beta)\boldsymbol{I}\right)^{-1}\left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \beta\hat{\boldsymbol{w}}\right). \quad (7)$$

By assuming $\lambda = \alpha + \beta$, (7) is re-written as

$$\boldsymbol{\mu}_w^* = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu} + \beta\left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \lambda\boldsymbol{I}\right)^{-1}\hat{\boldsymbol{w}}. \quad (8)$$

Moreover, by using the Woodbury identity[1] the inverse in the second term of (8) takes the form

$$\left(\boldsymbol{\Phi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^\top + \lambda\boldsymbol{I}\right)^{-1} = \frac{1}{\lambda}\left[\boldsymbol{I} - \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Phi}^\top\right] \quad (9)$$

which plugged back into (8) results in

$$\boldsymbol{\mu}_w^* = \underbrace{\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1}}_{(\boldsymbol{\Phi}^\top)^\dagger}\boldsymbol{\mu} + \frac{\beta}{\lambda}\left[\boldsymbol{I} - \underbrace{\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Phi}^\top}_{(\boldsymbol{\Phi}^\top)^\dagger}\right]\hat{\boldsymbol{w}}.$$
$$(10)$$

Equation (10) corresponds to a typical least squares solution with null space where for a linear system $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A}$ we

[1] $(\boldsymbol{A} + \boldsymbol{C}\boldsymbol{B}\boldsymbol{C}^\top)^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{C}\left(\boldsymbol{B}^{-1} + \boldsymbol{C}^\top \boldsymbol{A}^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{C}^\top \boldsymbol{A}^{-1}$

have as solution $\hat{Y} = \hat{X}A^* = \hat{X}\left(X^\dagger Y + Nv\right)$ where v satisfies a secondary goal and $N = I - X^\dagger X$.

For the remainder of the discussion we will assume, without loss of generality, $\frac{\beta}{\lambda} = 1$. For reference, this assumption is common in robot control when damped least squares and null space projectors are combined, see [23], [24].

B. Kernelizing the null space solution

As for the original solution of KMP, the optimal solution $\boldsymbol{\mu}_w^*$ is applied to a new test point to predict the expectation of the output as

$$
\begin{aligned}
\mathbb{E}\left[\boldsymbol{\xi}(s^*)\right] &= \boldsymbol{\Phi}(s^*)^\top \boldsymbol{\mu}_w^* \\
&= \boldsymbol{\Phi}(s^*)^\top \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu} + \Big[\boldsymbol{\Phi}(s^*)^\top \\
&\quad - \boldsymbol{\Phi}(s^*)^\top\boldsymbol{\Phi}\left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Phi}^\top\Big]\hat{w}.
\end{aligned} \tag{11}
$$

Using the *kernel trick* we can write

$$
\begin{aligned}
\mathbb{E}\left[\boldsymbol{\xi}(s^*)\right] &= k^*\left(K + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu} \\
&\quad + \left[\boldsymbol{\Phi}(s^*)^\top - k^*\left(K + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Phi}^\top\right]\hat{w}.
\end{aligned} \tag{12}
$$

For the sake of the derivation, we convert the null space desired weights $\hat{w}$ into a desired set of points in trajectory space. From the assumption of a parametric trajectory $\boldsymbol{\xi}(s) = \boldsymbol{\Phi}(s)^\top w$ we define an arbitrary number P of desired secondary target points as $\hat{\boldsymbol{\xi}}_1 = \boldsymbol{\Phi}(s_{p=1})^\top\hat{w}, \ldots, \hat{\boldsymbol{\xi}}_P = \boldsymbol{\Phi}(s_{p=P})^\top\hat{w}$ or, in matrix form,

$$
\hat{\boldsymbol{\xi}} = \hat{\boldsymbol{\Phi}}^\top\hat{w}. \tag{13}
$$

Next we approximate the optimal values of $\hat{w}$ given the desired secondary target points $\hat{\boldsymbol{\xi}}$ using the right pseudo-inverse of $\hat{\boldsymbol{\Phi}}^\top$, i.e.

$$
\hat{w} = \hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}}\right)^{-1}\hat{\boldsymbol{\xi}}. \tag{14}
$$

Plugging (14) back in (12) yields

$$
\begin{aligned}
\mathbb{E}[\boldsymbol{\xi}(s^*)] &= k^*\left(K + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu} + \Big[\boldsymbol{\Phi}(s^*)^\top \\
&\quad - k^*\left(K + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Phi}^\top\Big]\hat{\boldsymbol{\Phi}}\left(\hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}}\right)^{-1}\hat{\boldsymbol{\xi}}
\end{aligned} \tag{15}
$$

$$
= k^*\boldsymbol{\Psi}\boldsymbol{\mu} + \left[\hat{k}^* - k^*\boldsymbol{\Psi}\hat{K}\right]\left(\underline{K}\right)^{-1}\hat{\boldsymbol{\xi}}.
$$

with $\hat{k}^* = \boldsymbol{\Phi}(s^*)^\top\hat{\boldsymbol{\Phi}}$, $\hat{K} = \boldsymbol{\Phi}^\top\hat{\boldsymbol{\Phi}}$, $\underline{K} = \hat{\boldsymbol{\Phi}}^\top\hat{\boldsymbol{\Phi}}$ and $\boldsymbol{\Psi} = \left(K + \lambda\boldsymbol{\Sigma}\right)^{-1}$.

Finally, for a single desired secondary target and a squared-exponential kernel we have $\underline{K} = I$ and (15) becomes[2]

$$
\mathbb{E}[\boldsymbol{\xi}(s^*)] = k^*\boldsymbol{\Psi}\boldsymbol{\mu} + \left[\hat{k}^* - k^*\boldsymbol{\Psi}\hat{K}\right]\hat{\boldsymbol{\xi}}. \tag{16}
$$

Notice that (16) corresponds to (5) with the additional null space term $\left[\hat{k}^* - k^*\boldsymbol{\Psi}\hat{K}\right]\hat{\boldsymbol{\xi}}$. Equation (16) thus modulates

---

[2]To contrast our solution with the 'classical' KMP [14], throughout the rest of the paper we refer to (16) as the *null space KMP* (NS-KMP). In addition, we refer to $\hat{\boldsymbol{\xi}}$ as the *null space reference*.
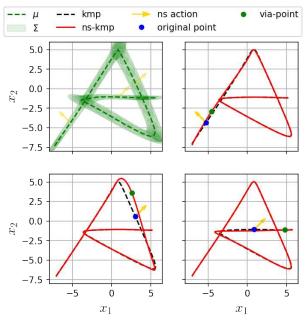

Fig. 1: Effect of the null space projector in modulating the original trajectory. *Top-left:* reference distribution with mean shown as a dashed curve and covariance as ellipses. *Top-right and bottom:* Modulation examples at three different points. A null space reference with the same magnitude across all plots is applied in the direction of the yellow arrow at the blue point. The non-modulated KMP trajectory (dashed) changes its shape (red) depending on the covariance at the point where the reference is applied. The green point is the via-point one would need to define to achieve the same modulation using classical KMP adaptation (see Section IV-C).

the expectation (5) in a computationally efficient manner. We leverage this property by modifying $\hat{\boldsymbol{\xi}}$ such that variations of the original trajectory are achieved, similarly to what is done in robot null space exploration [26]–[28]. However, unlike the null spaces found in robot control which are often strict (see [24] for an overview) the KMP null space possesses a set of properties derived from its probabilistic foundations that dictate how $\hat{\boldsymbol{\xi}}$ modulates (5).

IV. PROPERTIES OF THE KERNELIZED NULL SPACE PROJECTOR

A. Idempotence and 'soft' null space

A general property of null space projectors is idempotence, i.e. for a null space projector N we have $NN = N$ [25]. The projector derived in Section III-A is not guaranteed to have this property since, generally, $\left(I - \left(\boldsymbol{\Phi}^\top\right)^\dagger\boldsymbol{\Phi}^\top\right)\left(I - \left(\boldsymbol{\Phi}^\top\right)^\dagger\boldsymbol{\Phi}^\top\right) \neq \left(I - \left(\boldsymbol{\Phi}^\top\right)^\dagger\boldsymbol{\Phi}^\top\right)$. It should be noted, however, that as $\lambda\boldsymbol{\Sigma} \to 0$ idempotence is fulfilled, which can be verified trivially. In practice, this means that the larger $\boldsymbol{\Sigma}$, the more the null space reference is allowed to deform the original trajectory. For this reason we refer to it as a *soft* null space projector. In robotics, this is known to happen when computing null space projectors using the regularized pseudo-inverse of the Jacobian matrix. Indeed, in those cases, the null space tasks are known to affect the precision of the main one [24].
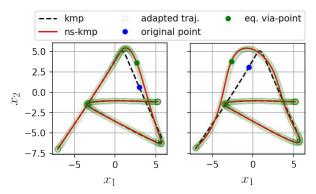
Fig. 2: Equivalence between null space modulation and via-point adaptation. A null space reference $\hat{\boldsymbol{\xi}}$ applied at input $\hat{s}$ modulates the original KMP (dashed), computed from (5), generating the red line, computed from (16). In the dashed line, $\hat{s}$ resulted in the blue point. In the red line $\hat{s}$ resulted in the green point. By treating the latter as a via-point (see Section II), classical KMP via-point adaptation (green circled line) produces the same result.

B. Covariance-weighted null space modulation

Unlike the robot control case, where regularization typically appears as a diagonal matrix, $\boldsymbol{\Sigma}_n$ are full covariance matrices. As a consequence, a null space reference generally does not affect all dimensions of the trajectory equally. The covariance at a given point influences both the magnitude and the directions along which a null space reference can modulate the trajectory. Figure 1 illustrates this property. The top-left image shows a trajectory distribution in 2D computed from examples of drawing a letter 'A' [29]. That distribution was used as a reference trajectory distribution in a KMP. Notice how the covariance changes throughout the trajectory. In the top-right and bottom images, we show the effect of applying a null space reference (yellow arrow), with the same magnitude, at different points along the trajectory. The dashed line is computed from (5) and the solid, red one from (16). The covariance in the null space projector constrains the modulated trajectory to deform less (more) along the directions where the variance is lower (higher). Additionally the modulation is applied along the axes of $\boldsymbol{\Sigma}_n$. This provides a principled way to use the variations in the original dataset to design optimal exploration strategies. Moreover, unlike typical exploration strategies that require the re-computation of model parameters or computationally costly operations to include new points in the trajectory, here adding new points is done inexpensively with (16). Indeed, the parameter $\boldsymbol{\Psi}$, that contains a potentially large matrix inverse operation, is computed only once from the reference trajectory distribution.

C. Equivalence to classical KMP adaptation

An important final property of NS-KMP is that null space modulation is equivalent to classical KMP modulation [14] given the appropriate via-point. For the sake of the explanation let us consider an input $\hat{s}$ where a null space reference $\hat{\boldsymbol{\xi}}$ was applied, resulting in an observation $\boldsymbol{\xi}(\hat{s}) \neq \hat{\boldsymbol{\xi}}$ after the null space projector. We have concluded empirically that by treating the pair $\{\hat{s}, \boldsymbol{\xi}(\hat{s})\}$ as a via-point in the classical KMP framework [14], and adding it to the reference

trajectory distribution with a low covariance, generates the same trajectory as the null-space-generated one. Figure 2 illustrates this property. This result suggests that NS-KMP is, indeed, a computationally-efficient tool for via-point-based motion adaptation.

V. EXPERIMENT I

We first report results on the re-planning of a 2D trajectory, learned from demonstrations, after an obstacle appears. We tackle this problem using null space exploration (NS-KMP) or by adding via-points (KMP, ProMP) to the demonstrated trajectory distribution. We compare the time performance of KMP, NS-KMP and ProMP.
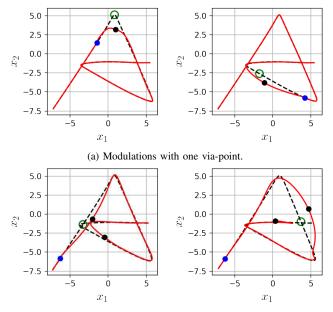
A. Setup

We use the handwritten letter dataset available in [29], particularly the dataset for letter 'A', to generate a trajectory distribution from demonstrations. It contains 10 demonstrations with time t and 2D position (x_1, x_2), which we use to train a mixture model with 8 components and retrieve a reference trajectory distribution $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{100}$ using Gaussian mixture regression. Moreover $s = t$ and $\boldsymbol{\xi} = [x_1 \ x_2]^\top$. In both KMP and NS-KMP implementations we use hyperparameters $\lambda = 0.1$, $l = 0.125$ while for ProMP we use 20 basis functions. The implementation was done in Python 3.8 running on an 8-core 11-Gen Intel Core i7-11850H @ 2.50GHz processor.

We use the learned models to simulate a robot planning a trajectory within a finite-horizon. At time t, the robot uses KMP, NS-KMP or ProMP to compute a trajectory for the horizon $\{t, \ldots, T\}$. We use $T = 1$ with $10ms$ increments between time steps. At a randomly chosen time instant in the horizon, an obstacle (a circle with radius 0.5) with position uniformly sampled from $\{\boldsymbol{\mu}_n\}_{n=1}^N$ appears (only once) along the planned path. The robot then tests alternative paths looking for one that does not collide with the obstacle. In NS-KMP, this is done by sampling values of input $\hat{s}$ and null space reference $\hat{\boldsymbol{\xi}}$ and computing the corresponding new trajectory using (16). The sampling interval for the input is $[t, \ldots, \min(t + 0.2, T - t)]$ and $[-1000, 1000]$ for the two dimensions of $\hat{\boldsymbol{\xi}}$ (we discuss this range of values in Section VII). In KMP and ProMP, finding alternative paths is done by sampling via-points in the neighborhood of the obstacle and inputs in the same range as for NS-KMP. To efficiently perform the exploration, including sampling null space references and via-points we used TPE [30], particularly the implementation available in the Optuna package [31]. We define a simple objective function to minimize $f(\boldsymbol{X}_{pred}) = c_{collision}(\boldsymbol{X}_{pred}) + c_{cont}(\boldsymbol{X}_{pred})$ where $\boldsymbol{X}_{pred}$ is the sequence of 2D points in the planned trajectory and $c_{collision}$ and c_{cont} are collision and continuity terms that receive values of 1000 if a collision happens in $\boldsymbol{X}_{pred}$ or the norm between any two consecutive points in $\boldsymbol{X}_{pred}$ exceeds the radius of the obstacle and 0 otherwise. The optimizer returns the first solution with zero cost.

B. Results using one via-point

Figure 3a shows example trajectories obtained from NS-KMP. We observe that the obstacle is avoided correctly in

(a) Modulations with one via-point.

(b) Modulations with two via-points.

Fig. 3: Examples of trajectories obtained using NS-KMP for different numbers of via-points. The blue point show the robot position when the obstacle (green circle) appears while the black point depicts the via-point(s) discovered by NS-KMP, which result in an adapted trajectory (red). The dashed line is the nominal trajectory encoded in the KMP.

TABLE I: Time to find solutions in 2D trajectory modulation.

	KMP	NS-KMP (ours)	ProMP
Prediction (ms)	67.4 ± 4.6	1.12 ± 0.33	0.24 ± 0.02
1 via-point (s)	1.62 ± 1.14	0.05 ± 0.04	0.02 ± 0.04
2 via-point (s)	-	1.70 ± 1.40	5.63 ± 6.20

VI. EXPERIMENT II

In a second experiment we investigate the ability of NS-KMP to modulate trajectories generated from multi-dimensional inputs. We consider a handover task where the end-effector position of one robot is computed from the one of another robot and adapted after an obstacle appears.

A. Setup

In order to train a NS-KMP we used the dataset collected from handover demonstrations in [32], with 5 demonstrations in the form of human hand and robot end-effector positions $\boldsymbol{x}_H$ and $\boldsymbol{x}_R$. The NS-KMP thus uses $s = \boldsymbol{x}_H$ and $\boldsymbol{\xi} = \boldsymbol{x}_R$. In order to simulate the human-robot handover task, the data was rescaled to fit the robot workspaces in the *pybullet* setup depicted in Fig. 4. As illustrated in Fig. 4, the left robot plays the role that the human played in the demonstrations by picking up an object (yellow box) and handing it over to the robot on the right. The latter starts with a planned motion (black line) computed using as input the trajectory planned by the left robot (unavailable in a real scenario with a human, only shown for illustration). In each trial the motion is interrupted by a cube-shaped obstacle with an axial diagonal of $10cm$ that appears randomly (both in time and location) once along the black line. All the NS-KMP parameters were the same as in V. The optimization framework was also the same, however $\hat{\boldsymbol{\xi}}$ was sampled from the range $[-2000, 2000]$ and the inputs (i.e. the left robot end-effector position) were sampled uniformly from the ones in the reference trajectory distribution. As in [14], we do not use ProMP as baseline here due to the difficulty in parameterizing basis functions for multi-dimensional inputs. Instead, we consider the optimization of Gaussian components in GMM as a baseline [6]. Specifically, we optimize the mean of one randomly chosen Gaussian until the collision cost is zero. We do not evaluate classical KMP in this section as the arguments that caused it to underperform in V-B are still valid (see also the analysis of computational complexity in Section VII-B).

B. Results

In 20 trials, the average time to compute a solution with NS-KMP after the obstacle appears was $0.16s \pm 0.13$. Fig. 4, *top-right* and *bottom*, shows original and adapted trajectories. The GMM baseline yielded $0.616s \pm 0.571$.

VII. DISCUSSION

A. Analysis of experimental results

The results in Section V revealed that our solution, despite not outperforming ProMP ($\approx 5\times$ faster at predicting), computes trajectories given a new desired point at the millisecond level (Table I, first row) with a non-optimized Python

different segments of the trajectory. Table I summarizes the time required by KMP, NS-KMP and ProMP to predict a trajectory from one null space reference or via-point (first row) and to find a solution to avoid the obstacle (second row). Each entry corresponds to the mean and standard deviation of 40 points. In the case of NS-KMP one *prediction* corresponds to computing $\hat{\boldsymbol{k}}^*$, $\hat{\boldsymbol{K}}$ and (16). For KMP it is the time of computing new matrices $\boldsymbol{K}$, $\boldsymbol{k}^*$ and (5), including the inversion. For ProMP it is the time to evaluate the basis functions at the new input, recompute the weight distribution and use it to compute a new trajectory (see [8]). Note that in this experiment we did not allow obstacles to appear at points where the trajectory intersects with itself (see these cases in Fig. 3b) since they require two via-points. The *prediction* times show that the parametric model ProMP was the fastest, closely followed by our non-parametric NS-KMP. Moreover, we observed comparable times between NS-KMP and ProMP to find a good trajectory (second row, *1 via-point (s)*), with KMP (first column) taking two orders of magnitude longer.

C. Results for collision avoidance with two via-points

Finally we investigated modulation by more than one via-point. For this we focused on the cases where the obstacle appears at places where the trajectory intersects with itself (Fig. 3b). Given the high computational cost of KMP shown in V-B we compare NS-KMP and ProMP only. Figure 3b shows that NS-KMP is capable of avoiding obstacles by modulating the planned trajectory with two via-points. As seen in Table I, last row, NS-KMP and ProMP achieved times within the same order of magnitude, with a slight disadvantage to ProMP.
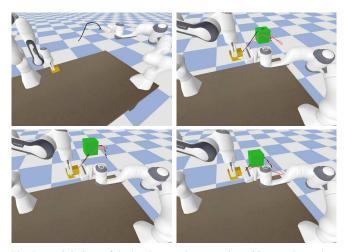
Fig. 4: *Top-left:* Start of the handover task. *Top-right and bottom:* Reproductions in simulation. The right robot motion is governed by the position of the left robot end-effector. For illustration, the black line shows the nominal trajectory of the right arm based on the planned motion of the left arm. An obstacle (green) appears during motion. Using NS-KMP the right arm replans its motion resulting in the trajectory in orange.

implementation. We believe this to be a noteworthy result for a non-parametric approach. Moreover, ProMP and NS-KMP performed at a similar level in finding a 1-via-point solution. However, ProMP is only $\approx 2.5\times$ faster, against $5\times$ for the individual prediction, suggesting that ProMP needs to try more via-points than NS-KMP to find a solution. This could be related to our choice of sampling via-points for ProMP around the obstacle, which may not be as efficient as the covariance-weighted exploration by NS-KMP. The reason for such choice is that we observed that the trajectories generated by ProMP often had arbitrary shapes (seriously violating the structure of the training data) if the search space of the via-point is not bounded or is large. The same reasoning applies to the 2-via-point case where NS-KMP took less time to find successful via-points.

Adaptation of learned multi-dimensional-input skills was a novelty introduced by KMP [14] and the results in Section VI show that NS-KMP provides a fast solution to adapting this type of skills. The GMM baseline was slower and the resulting trajectories were prone to distortions since only one Gaussian was being optimized at a time. The presented times for GMM can thus be seen as a lower bound since optimizing further parameters would require higher amounts of time. Despite the successful adaptation with NS-KMP we verified that, for our choice of hyperparameters, the time to compute a successful trajectory increases when the object appears closer to the handover location due to the decrease in variance (the null space 'hardens').

Some additional practical aspects of NS-KMP are worth discussing. As described in V–VI the magnitude of $\hat{\boldsymbol{\xi}}$ could be as high as 1000 or 2000 depending on the problem. Due to the probabilistic nature of the *soft* null space projector, one does not know *a priori* the required magnitude of $\hat{\boldsymbol{\xi}}$. It should thus be treated as a hyperparameter to be tuned depending on the problem. Additionally, while covariance-weighted null space exploration seems to alleviate the need

to manually define exploration strategies, in some scenarios one might need to violate the priors imposed by the data, e.g. if the object has a much larger size. In other words it may be desirable to 'soften' the null space. This can be achieved by manually increasing $\lambda\boldsymbol{\Sigma}$, particularly in the neighborhood of the concerned region.

B. Analysis of computational complexity

Classical KMP [14] predicts the expectation of $\boldsymbol{\xi}(\boldsymbol{s}^*)$ using (5). When a new desired point described by the mean $\bar{\boldsymbol{\mu}}$ and covariance $\bar{\boldsymbol{\Sigma}}$ is needed, the reference trajectory distribution $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^N$ is concatenated with $\{\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\}$, leading to an extended reference trajectory with length $N+1$. Consequently, KMP needs to recompute the term $(\boldsymbol{K} + \lambda\boldsymbol{\Sigma})^{-1}$ since the updated $\boldsymbol{\Sigma}$ includes $\bar{\boldsymbol{\Sigma}}$, see (3). Note that the dimension of $\boldsymbol{K}$ and $\boldsymbol{\Sigma}$ is $D(N+1) \times D(N+1)$ and therefore the computational complexity of classical KMP is $\mathcal{O}(D^3(N+1)^3)$. Recall that in (16) we only search in the null space, where $\boldsymbol{K}$ remains the same and therefore $\boldsymbol{\Psi}$ can be computed beforehand in an offline fashion. Therefore, the computation complexity of (16) is $\mathcal{O}(D^3 N^2)$, which is significantly faster than classical KMP.

VIII. CONCLUSION

We presented a formulation of kernelized movement primitives with a *soft* null space projector (NS-KMP). Such projector allows a secondary goal to modulate a primary one, in this case a learned trajectory. This modulation occurs in proportion to the covariance in the data. We leveraged the theoretical properties of this projector, as well as its computational efficiency, in letter-writing and handover tasks to quickly discover demonstration-guided, collision-free motions. NS-KMP, as a non-parametric approach, outperformed classical KMP in terms of computational complexity and performed competitively against the parametric approach ProMP. Moreover, we showed that NS-KMP provides fast adaptation in a fraction of a second in tasks where multi-dimensional input signals drive the robot behavior – an important part of human-robot interaction.

In future work we will leverage the scalability of NS-KMP to solve adaptation problems in higher-dimensional settings, such as humanoid robot control. We will also investigate how to integrate more diversified types of multi-dimensional inputs into the NS-KMP framework such as EMG signals [33]. Finally, we will investigate similar null space projectors for Gaussian processes.

References

[1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2002, pp. 1398–1403.

[2] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, vol. 37, no. 2, pp. 286–298, 2007.

[3] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, January 2016.

[4] Y. Huang, J. Silvério, L. Rozo, and D. G. Caldwell, "Generalized task-parameterized skill learning," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2018, pp. 5667–5474.

[5] F. Stulp and O. Sigaud, "Robot skill learning: From reinforcement learning to evolution strategies," *Paladyn, Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013.

[6] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007.

[7] M. Saveriano, F. Franzel, and D. Lee, "Merging position and orientation motion primitives," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2019, pp. 7041–7047.

[8] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2616–2624.

[9] R. A. Shyam, P. Lightbody, G. Das, P. Liu, S. Gomez-Gonzalez, and G. Neumann, "Improving local trajectory optimisation using probabilistic movement primitives," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 2666–2671.

[10] T. Löw, T. Bandyopadhyay, J. Williams, and P. V. Borges, "PROMPT: Probabilistic motion primitives based trajectory planning." in *Proc. Robotics: Science and Systems (R:SS)*, Virtual, July 2021.

[11] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[12] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2011, pp. 4569–4574.

[13] F. Frank, A. Paraschos, P. van der Smagt, and B. Cseke, "Constrained probabilistic movement primitives for robot trajectory adaptation," *IEEE Trans. on Robotics*, vol. 38, no. 4, pp. 2276–2294, 2022.

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

[15] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer, 2006.

[16] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, "Phase estimation for fast action recognition and trajectory generation in human–robot collaboration," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1579–1594, 2017.

[17] Y. Cui, J. Poon, J. V. Miro, K. Yamazaki, K. Sugimoto, and T. Matsubara, "Environment-adaptive interaction primitives through visual context for human–robot motor skill learning," *Autonomous Robots*, vol. 43, no. 5, pp. 1225–1240, 2019.

[18] J. Silvério, Y. Huang, L. Rozo, S. Calinon, and D. G. Caldwell, "Probabilistic learning of torque controllers from kinematic and force constraints," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8.

[19] J. Silvério, Y. Huang, L. Rozo, and D. G. Caldwell, "An uncertainty-aware minimal intervention control strategy learned from demonstrations," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2018, pp. 6065–6071.

[20] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, MA, USA: MIT Press, 2006.

[21] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics 'Robots in Unstructured Environments'*, 1991, pp. 1211–1216.

[22] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[23] R. Mayorga, K. Ma, and A. Wong, "A robust local approach for the obstacle avoidance of redundant robot manipulators," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, vol. 3, 1992, pp. 1727–1734.

[24] A. S. Deo and I. D. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *Journal of Intelligent and Robotic Systems*, vol. 14, no. 1, pp. 43–68, 1995.

[25] A. Dietrich, C. Ott, and A. Albu-Schäffer, "An overview of null space projections for redundant, torque-controlled robots," *The International Journal of Robotics Research*, vol. 34, no. 11, pp. 1385–1400, 2015.

[26] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, 2008, pp. 91–98.

[27] Y. Huang, J. Silvério, L. Rozo, and D. G. Caldwell, "Hybrid probabilistic trajectory optimization using null-space exploration," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018, pp. 7226–7232.

[28] M. Schneider and W. Ertel, "Robot learning by demonstration with local Gaussian process regression," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 255–260.

[29] S. Calinon and D. Lee, *Learning Control*. Dordrecht: Springer Netherlands, 2019, pp. 1261–1312.

[30] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems (NIPS)*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011.

[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[32] J. Silvério, Y. Huang, F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Uncertainty-aware imitation learning using kernelized movement primitives," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2019, pp. 90–97.

[33] J. Vogel, A. Hagengruber, M. Iskandar, G. Quere, U. Leipscher, S. Bustamante, A. Dietrich, H. Hoppner, D. Leidner, and A. Albu-Schaffer, "EDAN: An EMG-controlled daily assistant to help people with physical disabilities," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2020, pp. 4183–4190.