This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

Fast Non-Parametric Heteroscedastic Imitation Learning With Geometric Priors

Maximilian Mühlbauer[1,2], Arne Sachtler[1,2], Markus Knauer[2,1], Cem Küçükgenç[1,2], Yanlong Huang[3], Alin Albu-Schäffer[2,1], João Silvério[2]

***Abstract*—When learning probabilistic policies from human demonstrations, data-efficient learning and fast adaptations to new scenarios are key requirements. One popular way to achieve intuitive and reliable adaptations is through non-parametric, typically kernel-based, methods. However, existing solutions either fail to account for the geometry of manifolds common in robotics, limiting data efficiency, or, when geometry-aware, provide unreliable uncertainty estimates or require retraining to adapt. We propose a non-parametric approach leveraging geometric priors in scenarios of data scarcity and heteroscedastic uncertainties for probabilistic modeling. We utilize the method to formulate policies based on time or robot state, where non-separable diagonal kernels allow capturing uncertainty relations between degrees of freedom for same-sized in- and outputs. Fast updates, requiring less than $3\,\mathrm{ms}$ for a trajectory involving both position and orientation are possible through an optimized formulation. Our approach supports both manifold-valued input and manifold-valued output with large orientation changes. Using task parameterization, adaptation to different object poses is easily possible. We evaluate the approach on a set of toy examples and on real robot manipulation tasks both in autonomous execution and in shared control scenarios.**

***Index Terms*—Human-Robot Interaction, Imitation Learning, Uncertainty Quantification, Task and Motion Planning**

I. Introduction

Probabilistic representations of movement primitives are key in applications with requirements in robustness and safety guarantees, as for example in space robotics [1], underwater robotics [2] or healthcare assistance [3]. *Parametric* approaches such as Gaussian Mixture Models (GMMs) [4, 5] provide an *aleatoric* (inherent to the data) uncertainty estimate while *non-parametric* approaches like Gaussian Processes (GPs) [6] provide *epistemic* (lack of data) uncertainty [7]. Kernelized Movement Primitives (KMPs) [8] and their extensions [9, 10] have shown remarkable qualities regarding modeling capability and adaptability, while in parallel providing *both* uncertainty estimates [11]. Both uncertainty types are required in applications – the epistemic uncertainty estimate can, e.g., be used to render the robot compliant when unsure about its actions [12] while the aleatoric uncertainty helps to compute control gains encoding task variability [13]. Heteroscedasticity allows our method to model different uncertainties for different phases of a task, which is a key requirement in policy arbitration [14, 13]. KMPs have already been applied to a range of learning problems, such as manufacturing tasks [12], in space applications [1] or to guide reinforcement learning [15].

Currently, these approaches are however only designed for Euclidean [8, 10] or single tangent space (STS) [9], linearizing curved manifolds at one point, thus inducing distortions [16] and not allowing to incorporate prior geometric knowledge. In contrast, approaches for multiple tangent spaces as already developed for GMMs [4] induce much less distortion through the utilization of multiple linearization points. A notable exception for KMPs is developed in [17] fusing multiple STS KMPs [9] on multiple tangent spaces on $\mathrm{SO}(3)$, at the cost, however, of an intricate formulation and a lack of generalizability to other geometries. Additionally, each of the STS KMPs still suffers from issues resulting from using a single linearization point. Conversely, Lopez-Custodio et al. [18] developed a kernel-based, on-manifold approach which, however, uses different distributions per manifold which are hard to join in product manifolds and computationally slow. This approach also only outputs aleatoric uncertainty, neglecting epistemic uncertainty which is fundamental in many imitation learning problems. Our solution (Fig. 1) combines the advantages of the aforementioned methods, namely fast predictions with aleatoric and epistemic uncertainty. On selected manifolds such as $\mathrm{SO}(3)$, we achieve global validity through the double-coverage by $\mathcal{S}^3$. Using the Riemannian Gaussian distribution [4, 5], no discrete linearization points are required, removing the covariance distortions caused by STS methods (Section III).

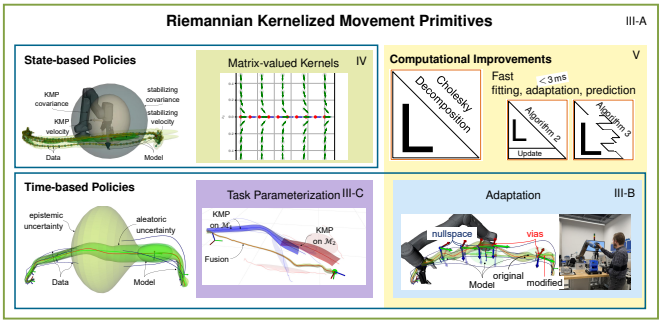


Fig. 1: Overview of our contributions. In Section III we leverage Riemannian geometry to learn, adapt and task-parameterize demonstrated movement primitives including both aleatoric and epistemic uncertainties. Section IV further extends the framework to DoF-dependent uncertainties. Finally, Section V improves the efficiency of the required computations.

Moreover, currently available kernel-based methods, using separable kernels, can only predict scalar-valued epistemic uncertainties as also seen in the decomposition in [12]. Recently, matrix-valued kernels have become available [19, 20] which allow for encoding correlations for same-sized inputs and outputs. As a large step in the direction of such kernels, we leverage diagonal kernels to improve the learning of state-based Dynamical Systems (DSs) (Section IV).

Finally, as in any non-parametric method, runtime in KMPs is a function of the number of data points and in general slower than for parametric methods. Especially adding or removing points is computationally expensive, requiring the inversion of a large matrix. Wang et al. [21] recently proposed a method for avoiding to recalculate the whole inverse. Similar to GPs [6], we leverage the Cholesky decomposition, significantly improving runtime over both vanilla KMP and [21] while keeping flexibility in adding and removing points (Section V).

In summary, the key contributions of our work are:

The research leading to those results has received funding from the Horizon Europe programme under grant agreements No.101136067 (INVERSE) and No.101298421 (GRAIL). Results were also partiall achieved within the ACOR project funded by the Federal Ministry BMWK (50RP2240A) and the DLR project "ASPIRO".

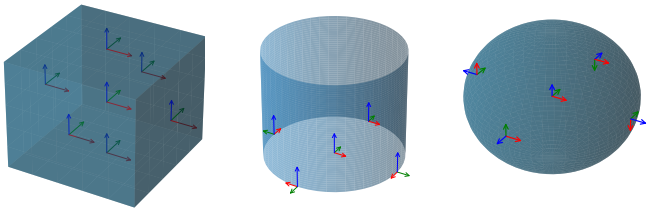
[1] Department of Computer Engineering, Technical University of Munich, Friedrich-Ludwig-Bauer-Str. 3, Garching, Germany.

[2] German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), Münchener Str. 20, 82234 Weßling, Germany.

[3] School of Computer Science, University of Leeds, Leeds, UK.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270



(a) $\mathcal{M}_1 \colon \mathbb{R}^3 \times \mathrm{SO}(3)$. (b) $\mathcal{M}_2 \colon \mathcal{S}^1 \times \mathbb{R}^2 \times \mathrm{SO}(3)$. (c) $\mathcal{M}_3 \colon \mathcal{S}^2 \times \mathbb{R}^1 \times \mathrm{SO}(3)$.

Fig. 2: Manifolds used in this work, proposed by [22]. Cylindrical ($\mathcal{M}_2$) or spherical ($\mathcal{M}_3$) coordinates allow expressing some tasks more data-efficient compared to Cartesian ($\mathcal{M}_1$) coordinates. Coordinate systems in each image depict the unit quaternion $(0, 0, 0, 1)^\top$ for different positions on the manifold.

1) A Riemannian formulation of a non-parametric approach to imitation learning [8] providing both aleatoric and epistemic heteroscedastic uncertainties (Section III-A);
2) The Riemannian derivation of two adaptation schemes for different requirements based on via points [8] as well as on a soft null space projector [10] (Section III-B);
3) Task-parameterization of the method on different charts [22] on $\mathrm{SE}(3)$ (Section III-C);
4) The integration of non-separable matrix-valued kernels in the approach to allow for the prediction of a matrix-valued epistemic uncertainty (Section IV-B);
5) Computational improvements allowing for fast prediction and adaptation in several geometries commonly used in robotics problems (Section V).

At the end of Sections III to V we provide toy examples evaluating the key contributions in simplified scenarios. Furthermore, we have implemented both time-based policies and their modification as well as state-based DSs on a torque-controlled robot (Section VI). Section II covers background. We discuss results in Section VII and conclude in Section VIII.

II. Background

A. Riemannian Manifolds, Optimization and Statistics

Many quantities in robotics, such as orientations in $\mathrm{SO}(3)$, cannot be modeled using Euclidean geometry. Riemannian manifolds allow for a principled treatment of such non-Euclidean geometries [5]. Common to all smooth manifolds $\mathcal{M}$ is that the logarithm function $\boldsymbol{u}_{12} = \mathrm{Log}_{\boldsymbol{x}_1}^{\mathcal{M}}(\boldsymbol{x}_2)$ maps a point $\boldsymbol{x}_2 \in \mathcal{M}$ onto the tangent space $\mathcal{T}_{\boldsymbol{x}_1}\mathcal{M}$ of $\mathcal{M}$ at $\boldsymbol{x}_1$, with the exponential function locally performing the inverse, calculating $\boldsymbol{x}_2 = \mathrm{Exp}_{\boldsymbol{x}_1}^{\mathcal{M}}(\boldsymbol{u}_{12})$. *Parallel transport* by the Levi-Civita connection moves tangent vectors between tangent spaces at different points on $\mathcal{M}$. Optimization on manifolds can be performed using geodesic regression [23] (multivariate case: [24]), making use of *parallel transport* to approximate a small update in the vicinity of the current linearization point.

We implement geometric priors by choosing different Riemannian metrics on $\mathrm{SE}(3)$, which we construct as *pullback metrics* from charts of different *representational manifolds* $\mathcal{M}_i$ as shown in Fig. 2. All manifolds $\mathcal{M}_i$ are six-dimensional product manifolds, and we construct maps $\mathcal{F}_i : \mathrm{SE}(3) \to \mathcal{M}_i$ that relate poses $\boldsymbol{x} \in \mathrm{SE}(3)$ to their representations $\boldsymbol{p} \in \mathcal{M}_i$, where the Jacobians $\boldsymbol{J}_{\mathcal{F}_i}$ as given in [13] push tangent vectors between tangent spaces of $\mathcal{M}_i$. Further, we have charts $\varphi_i : \mathcal{U} \subset \mathcal{M}_i \to \varphi_i(\mathcal{U}) \subset \mathbb{R}^6$ for the representational manifolds, which can be constructed by stacking charts of the individual factor manifolds due to the product manifold structure of $\mathcal{M}_i$. Using the concatenations $\psi_i = \varphi_i \circ \mathcal{F}_i$, we obtain charts $\psi_i : \mathcal{U}_i \subseteq \mathrm{SE}(3) \to \psi_i(\mathcal{U}_i) \subset \mathbb{R}^6$, with which we can pull the Euclidean metric of $\mathbb{R}^6$ back to $\mathrm{SE}(3)$ (first fundamental form). This induces different Riemannian metrics on $\mathrm{SE}(3)$ for each choice of *representational manifold* $\mathcal{M}_i$, and an ordered basis for tangent spaces $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}_i$.

The Riemannian Gaussian distribution [4, 5] uses geodesic distances computed via the log map based on this metric

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}}^{\mathcal{M}_i}(\boldsymbol{x})^\top \boldsymbol{\Sigma}^{-1} \mathrm{Log}_{\boldsymbol{\mu}}^{\mathcal{M}_i}(\boldsymbol{x})}, \tag{1}$$

with mean $\boldsymbol{\mu} \in \mathcal{M}_i$, covariance $\boldsymbol{\Sigma} \in \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}_i \otimes \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}_i$ and the dimension D of $\mathcal{M}_i$. Using the metric induced by the charts defined above, we get different distances and thus different Gaussian shapes depending on the chosen manifold. As a limitation of the Riemannian Gaussian, (1) is only well-defined for covariances small enough that Exp is a diffeomorphism.

B. Probabilistic Learning from Demonstration

One key goal in probabilistic movement modeling, besides an accurate reconstruction, is to provide an estimate of uncertainty. GMMs [25, 5] modeling *aleatoric uncertainty* encode the joint distribution between input $\boldsymbol{s}$ and output $\boldsymbol{\xi}$ with M Gaussians and mixing coefficients π_m ($\sum_{m=1}^{M} \pi_m = 1$),

$$\begin{bmatrix}\boldsymbol{s}\\ \boldsymbol{\xi}\end{bmatrix} \sim \sum_{m=1}^{M} \pi_m \mathcal{N}\left(\begin{bmatrix}\boldsymbol{s}\\ \boldsymbol{\xi}\end{bmatrix} \middle| \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m\right). \tag{2}$$

The conditional distribution of $\boldsymbol{\xi}$ given $\boldsymbol{s}$ evaluates to

$$p(\boldsymbol{\xi}|\boldsymbol{s}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{\xi}|\boldsymbol{\mu}_{m|\boldsymbol{s}}, \boldsymbol{\Sigma}_{m|\boldsymbol{s}}) \tag{3}$$

using Gaussian Mixture Regression (GMR). The extension of GMM / GMR to Riemannian manifolds is given by [4, 5].

GPs [6] compute posterior mean and covariance of a function $\boldsymbol{\xi}(\boldsymbol{s}) = f(\boldsymbol{s}) + \epsilon$, modeling the *epistemic uncertainty*; an extension to Riemannian manifolds is given in [26]. Based on a probabilistic reference $\mathcal{D} = \{\boldsymbol{s}_n, \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{N}$ obtained through GMM / GMR, KMPs [8] model both *aleatoric* and *epistemic uncertainty*. For a test point $\boldsymbol{s}^*$, we obtain

$$\mathrm{E}[\boldsymbol{\xi}^*(\boldsymbol{s})] = \boldsymbol{K}(\boldsymbol{s}^*, \boldsymbol{s})\left(\boldsymbol{K}(\boldsymbol{s}, \boldsymbol{s}) + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\mu} \tag{4}$$

$$\mathrm{Cov}[\boldsymbol{\xi}^*(\boldsymbol{s})] = \alpha\Big(\boldsymbol{K}(\boldsymbol{s}^*, \boldsymbol{s}^*) - \boldsymbol{K}(\boldsymbol{s}^*, \boldsymbol{s})\left(\boldsymbol{K}(\boldsymbol{s}, \boldsymbol{s}) + \lambda_c\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{K}(\boldsymbol{s}, \boldsymbol{s}^*)\Big), \tag{5}$$

where $\boldsymbol{K}$ is composed of kernel matrices, $\boldsymbol{\mu} = [\boldsymbol{\mu}_1^\top, \ldots, \boldsymbol{\mu}_N^\top]^\top$ and $\boldsymbol{\Sigma} = \mathrm{blockdiag}(\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_N)$ are set from the probabilistic reference. λ, λ_c and α are regularization and scaling hyperparameters. By predicting full, input-dependent covariance matrices (5), KMPs are *heteroscedastic* models. Please see [8] for a detailed derivation. Note the connection to GPs [6], where we have $\lambda_c = \lambda$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$ (*homoscedasticity*), $\alpha = 1$ and $\boldsymbol{\mu}$ are directly the observations $\boldsymbol{\xi}(\boldsymbol{s})$.

III. Riemannian Kernelized Movement Primitives

We model the distribution of a random variable $\boldsymbol{\xi}(\boldsymbol{s})$

$$p(\boldsymbol{\xi}(\boldsymbol{s})|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\xi}(\boldsymbol{s})|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{6}$$

where $\boldsymbol{s}$ and $\boldsymbol{\xi}(\boldsymbol{s})$ are Euclidean or manifold-valued (Fig. 2) and $\boldsymbol{\Sigma}$ contains both aleatoric and epistemic uncertainty. In the following, we provide a derivation based on encoding a probabilistic reference $\mathcal{D} = \{\boldsymbol{s}_n, \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{N}$ with N entries of observations $\boldsymbol{\mu}_n \in \mathcal{M}$ with known covariance $\boldsymbol{\Sigma}_n \in \mathcal{T}_{\boldsymbol{\mu}_n}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\mu}_n}\mathcal{M}$ for predicting new outputs (III-A), the adaptation to via points (III-B), its task parameterization (III-C) as well as an evaluation in simplified yet representative scenarios (III-D).

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

A. Movement Primitive Prediction

As in [8, 9, 10] we assume a parametric function

$$\boldsymbol{\xi}(\boldsymbol{s}) = \mathrm{Exp}_{\boldsymbol{e}}^{\mathcal{M}}\Bigg(\underbrace{\begin{bmatrix}\boldsymbol{\phi}(\boldsymbol{s}) & \dots & \mathbf{0}\\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \boldsymbol{\phi}(\boldsymbol{s})\end{bmatrix}^{\top}}_{\boldsymbol{\Phi}(\boldsymbol{s})}\boldsymbol{w}\Bigg) = \mathrm{Exp}_{\boldsymbol{e}}\left(\boldsymbol{\Phi}(\boldsymbol{s})^{\top}\boldsymbol{w}\right) \tag{7}$$

where $\boldsymbol{\phi}(\boldsymbol{s}) \in \mathbb{R}^B$ is a B-dimensional vector of basis functions depending on $\boldsymbol{s}$. $\boldsymbol{w} \in \mathbb{R}^{BD}$ is a vector of parametric weights, with the tangent dimension D of $\boldsymbol{\xi}(\boldsymbol{s}) \in \mathcal{M}$ and $\boldsymbol{e} \in \mathcal{M}$ the process mean function value. Assuming normally distributed weights $\boldsymbol{w}$ with $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, the distribution in (7) is also normally distributed [8]. As in vanilla KMP, we seek to minimize the Kullback-Leibler divergence between reference and predictive distributions. To this end, we maximize the posterior probability by solving the regularized weighted least squares problem with residual vectors calculated in $\mathcal{T}_{\boldsymbol{\mu}_n}\mathcal{M}$, while $\boldsymbol{\mu}_w$ is expressed in $\mathcal{T}_{\boldsymbol{e}}\mathcal{M}$,

$$J(\boldsymbol{\mu}_w) = \sum_{n=1}^{N}\Big(\mathrm{Log}_{\boldsymbol{\mu}_n}\left(\mathrm{Exp}_{\boldsymbol{e}}\left(\boldsymbol{\Phi}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w\right)\right)^{\top}\boldsymbol{\Sigma}_n^{-1} \\ \mathrm{Log}_{\boldsymbol{\mu}_n}\left(\mathrm{Exp}_{\boldsymbol{e}}\left(\boldsymbol{\Phi}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w\right)\right)\Big) + \lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w. \tag{8}$$

Taking inspiration from [24], we approximate this nonlinear cost function by a quadratic and kernelizable problem

$$J(\boldsymbol{\mu}_w) \approx \sum_{n=1}^{N}\Big(\left(\Gamma_{\boldsymbol{\mu}_n}\left(\boldsymbol{\Phi}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w\right) + \mathrm{Log}_{\boldsymbol{\mu}_n}\boldsymbol{e}\right)^{\top}\boldsymbol{\Sigma}_n^{-1} \\ \left(\Gamma_{\boldsymbol{\mu}_n}\left(\boldsymbol{\Phi}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w\right) + \mathrm{Log}_{\boldsymbol{\mu}_n}\boldsymbol{e}\right)\Big) + \lambda\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w, \tag{9}$$

where $\Gamma_{\boldsymbol{\mu}_n}(\cdot)$ denotes the parallel transport $\Gamma_{\boldsymbol{e}\to\boldsymbol{\mu}_n}(\cdot)$ from the mean function value $\boldsymbol{e}$ to $\boldsymbol{\mu}_n$. Note that this cost function is never evaluated explicitly as the basis functions project into an infinite-dimensional feature space. Comparing the kernelized result with the function-space derivation of Riemannian KMPs in Appendix A-B shows that indeed, no precision is lost by this approximation. Unlike the weight-space derivation in [9], we do not linearize the problem in a single tangent space $\mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$ at an arbitrary $\boldsymbol{x}_0$. Instead, we can utilize Riemannian Gaussians with manifold-valued mean $\boldsymbol{\mu}_n$ and covariance expressed in $\mathcal{T}_{\boldsymbol{\mu}_n}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\mu}_n}\mathcal{M}$ as reference, e.g., obtained from a Riemannian Gaussian Mixture Regression [4]. We furthermore do not restrict the solution to quaternions, but remain general to Riemannian manifolds where the parallel transport $\Gamma_{\boldsymbol{e}\to\boldsymbol{\mu}_n}(\cdot)$ can be expressed using left-multiplication of a rotation matrix $\boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}_n}$. Differentiating the objective (9) and equating to zero thus results in

$$\Big(\left(\boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}}\boldsymbol{\Phi}(\boldsymbol{s})^{\top}\right)^{\top}\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}}\boldsymbol{\Phi}(\boldsymbol{s})^{\top}\right) + \lambda\boldsymbol{I}\Big)\boldsymbol{\mu}_w^{*} = \\ -\left(\boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}}\boldsymbol{\Phi}(\boldsymbol{s})^{\top}\right)^{\top}\boldsymbol{\Sigma}^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}), \tag{10}$$

where stacked matrices from the probabilistic reference

$$\boldsymbol{\Phi} = \left[\boldsymbol{\Phi}(\boldsymbol{s}_1)\dots\boldsymbol{\Phi}(\boldsymbol{s}_N)\right], \quad \boldsymbol{\mu} = \left[\boldsymbol{\mu}_1\dots\boldsymbol{\mu}_N\right]^{\top}, \\ \boldsymbol{\Sigma} = \mathrm{blockdiag}(\boldsymbol{\Sigma}_1\dots\boldsymbol{\Sigma}_N) \tag{11}$$

have been used. Using the matrix identity [27]

$$\left(\boldsymbol{B}^{\top}\boldsymbol{R}^{-1}\boldsymbol{B} + \boldsymbol{P}^{-1}\right)^{-1}\boldsymbol{B}^{\top}\boldsymbol{R}^{-1} = \boldsymbol{P}\boldsymbol{B}^{\top}(\boldsymbol{B}\boldsymbol{P}\boldsymbol{B}^{\top} + \boldsymbol{R})^{-1},$$

and setting $\boldsymbol{\Phi}_{\|}(\boldsymbol{s}_n) = \boldsymbol{\Phi}(\boldsymbol{s}_n)\boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}_n}^{\top}$ we arrive at the solution

$$\boldsymbol{\mu}_w^{*} = -\boldsymbol{\Phi}_{\|}(\boldsymbol{s})\left(\boldsymbol{\Phi}_{\|}(\boldsymbol{s})^{\top}\boldsymbol{\Phi}_{\|}(\boldsymbol{s}) + \lambda\boldsymbol{\Sigma}\right)^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}), \tag{12}$$

Substituting (12) in (7) and applying the *kernel trick* [27],

$$\mathbb{E}\left[\boldsymbol{\xi}(\boldsymbol{s}^{*})\right] = \mathrm{Exp}_{\boldsymbol{e}}\left(\boldsymbol{\Phi}_{\|}(\boldsymbol{s}^{*})^{\top}\boldsymbol{\mu}_w^{*}\right) \\ = \mathrm{Exp}_{\boldsymbol{e}}\left(-\boldsymbol{k}_{\|}^{*}\left(\boldsymbol{K}_{\|} + \lambda\boldsymbol{\Sigma}\right)^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e})\right) \tag{13}$$

follows for the expectation which is the same result as in Appendix A-B, where the KMP is derived as a Gaussian conditioning. Note that on the manifold $\mathbb{R}^n$, (13) evaluates to the same result as the Euclidean KMP prediction (4) - our Riemannian derivation thus contains the Euclidean variant as a special case. The covariance evaluates to (Appendix A-B)

$$\boldsymbol{\Sigma}(\boldsymbol{s}^{*}) = \boldsymbol{k}_{\|}(\boldsymbol{s}^{*},\boldsymbol{s}^{*}) - \boldsymbol{k}_{\|}^{*}\left(\boldsymbol{K}_{\|} + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{k}_{\|}^{*\top}. \tag{14}$$

Calculating the mean via (13) suffers from imprecision for certain manifolds when predicting a new point close to an antipodal point – e.g., on $\mathrm{SO}(3)$, tangent entries of $-\pi$ and π denote the same orientation change. We can often avoid these cases by recentering around the Fréchet mean of the data.

When predicting the orientation on $\mathrm{SO}(3)$, we can also make use of the double coverage of $\mathrm{SO}(3)$ by $\mathcal{S}^3$. Leveraging the whole range of $[-2\pi, 2\pi)$ for values in $\mathcal{T}\mathcal{S}^3$ instead of just $[-\pi, \pi)$ for $\mathcal{T}\mathrm{SO}(3)$, we can "push" the discontinuity between $-\pi$ and π away from query input. If the trajectory involves orientation changes of less than $360°$, one updated instance of $\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e})$ is sufficient. For larger orientation changes, we can create multiple instances where we always select the instance with the discontinuity furthest from the query input. The universal cover $\mathbb{R}^1$ of $\mathcal{S}^1$ similarly allows for resolving discontinuities on this manifold.

B. Movement Primitive Adaptation

Two main approaches to adapt KMPs to novel situations exist, suiting different types of use cases.

a) Via-point based adaptation: As in the original KMP [8], adaptation can be performed by extending the reference $\mathcal{D}$ by a *desired database* $\overline{\mathcal{D}} = \{\overline{\boldsymbol{s}}_p, \overline{\boldsymbol{\mu}}_p, \overline{\boldsymbol{\Sigma}}_p\}_{p=1}^{P}$. Low covariances $\overline{\boldsymbol{\Sigma}}_p$ result in a modification close to $\overline{\boldsymbol{\mu}}_p$, larger covariances in a prediction close to the original one. Augmenting the KMP with $\overline{\mathcal{D}}$ requires the extension of $\boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ and the costly recomputation of its inverse. In Section V, we provide computationally faster alternatives for calculating the update, allowing for a quick update, e.g., through human correction.

b) Modulation with uncertainty-aware deformations: Soft null space adaptations [10] guide the adaptation by the previously learned covariance; they have been used to constrain reinforcement learning [15] or coarse human modifications [3]. The goal of the adaptation [10] is to keep weights $\boldsymbol{\mu}_w$ in (9) close to desired weights $\hat{\boldsymbol{w}}$, which can be achieved by adding the penalty term $\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}})^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}})$ to (9). For a desired modification $\hat{\boldsymbol{\xi}} \in \mathcal{M}$, similar to the Euclidean derivations in [10], we can calculate a new expected value $\boldsymbol{\xi}'(\boldsymbol{s}^{*})$ (detailed in Appendix B-A), extending (7)

$$\mathbb{E}\left[\boldsymbol{\xi}'(\boldsymbol{s}^{*})\right] = \mathrm{Exp}_{\boldsymbol{e}}\Big(-\boldsymbol{k}_{\|}^{*}\boldsymbol{\Psi}_{\|}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}) + \left[\hat{\boldsymbol{k}}_{\|}^{*} - \boldsymbol{k}_{\|}^{*}\boldsymbol{\Psi}_{\|}\hat{\boldsymbol{K}}_{\|}\right] \\ \left(\underline{\boldsymbol{K}}_{\|}\right)^{-1}\Gamma_{\boldsymbol{e}\to\boldsymbol{\xi}(\boldsymbol{s}^{*})}\left(\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^{*})}(\hat{\boldsymbol{\xi}})\right)\Big) \tag{15}$$

where $\boldsymbol{\Psi}_{\|} = \left(\boldsymbol{K}_{\|} + \lambda\boldsymbol{\Sigma}\right)^{-1}$, $\hat{\boldsymbol{K}}_{\|} = \boldsymbol{\Phi}_{\|}^{\top}\hat{\boldsymbol{\Phi}}_{\|}$ and $\underline{\boldsymbol{K}}_{\|} = \hat{\boldsymbol{\Phi}}_{\|}^{\top}\hat{\boldsymbol{\Phi}}_{\|}$ and $\boldsymbol{\xi}(\boldsymbol{s}^{*})$ is the prediction before applying the modification. Owing to the uncertainty-aware deformation, $\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^{*})}(\hat{\boldsymbol{\xi}})$ is in practice scaled to achieve the desired deformation as in the Euclidean case [10]. In learning approaches such as [15], we can use $\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^{*})}(\hat{\boldsymbol{\xi}})$ as *action space* for a covariance-guided exploration.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

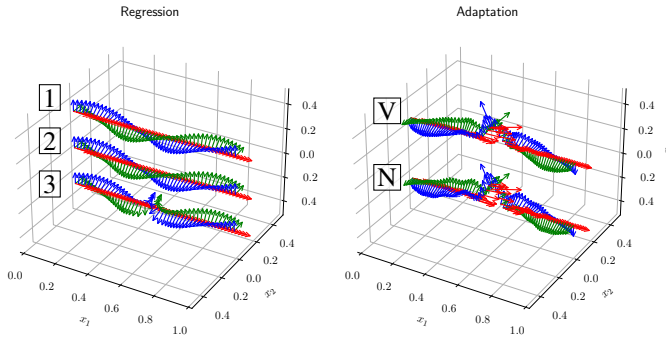


Fig. 3: *Left*: Pose data 1 reproduced by a Riemannian KMP (III-A) 2 with correct rotations. In contrast, the STS KMP [9] 3 suffers from incorrect rotations. *Right*: Adding a via point V 0.1 m above the trajectory with a relative rotation of -0.35 rad about the y axis and a null space action N. Via point and nullspace target are visualized by a larger coordinate system.

TABLE I: DIFFERENCES BETWEEN THE PREDICTION OF DIFFERENT METHODS AND GROUND TRUTH FOR THE SPIRAL TRAJECTORY (FIG. 3).

Method	Position Difference (mm)	Rotation Difference (mrad)
GMR (STS)	2.47 ± 1.02	3139.36 ± 11.47
KMP (STS) [9]	2.47 ± 1.03	269.36 ± 646.48
Riemannian GMR [4]	2.47 ± 1.09	15.27 ± 8.81
Riemannian KMP (ours)	$\mathbf{2.46 \pm 1.12}$	$\mathbf{15.15 \pm 9.05}$
Nadaraya-Watson [18]	3.05 ± 1.22	15.29 ± 9.15

C. Task Parameterization

Using task parameterization [25], we can learn movement models locally, with respect to different coordinate origins $\boldsymbol{x}_i \in \mathrm{SE}(3)$. Defining new values of these origins at execution time, e.g., due to new object positions/orientations, facilitates skill generalization. Such $\mathrm{SE}(3)$ transformations equate to a rotation $\boldsymbol{A} \in \mathbb{R}^{D\times D}$ of the tangent space and a translation $\boldsymbol{b} \in \mathcal{M}$ [4] with $\boldsymbol{\mu} \in \mathcal{M}$ and $\boldsymbol{\Sigma} \in \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$

$$\boldsymbol{\mu}_{\boldsymbol{A},\boldsymbol{b}} = \mathrm{Exp}_{\boldsymbol{b}}\left(\boldsymbol{A}\mathrm{Log}_{\boldsymbol{e}}\left(\boldsymbol{\mu}\right)\right), \tag{16}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{A},\boldsymbol{b}} = \boldsymbol{A}\boldsymbol{\Sigma}\boldsymbol{A}^{\top}. \tag{17}$$

Note that unlike in [4], the required parallel transport evaluates to identity on $\mathcal{M}_1$,[1] as tangent spaces are identified trivially on $\mathbb{R}^3$ respectively via the quaternion product on $\mathcal{S}^3$. Local models on $\mathcal{M}_1$ can be rotated by setting $\boldsymbol{A} = \mathrm{blockdiag}\left(\boldsymbol{R}_{\boldsymbol{eb}}, \boldsymbol{I}\right)$ corresponding to the rotation of position degrees of freedom (DoFs) in a $\mathrm{SE}(3)$ transformation and translated by the new mean $\boldsymbol{b}$. After transforming all Gaussian predictions into a common coordinate system by applying (16) – (17), the Riemannian Gaussian product [4] allows for fusing the prediction.

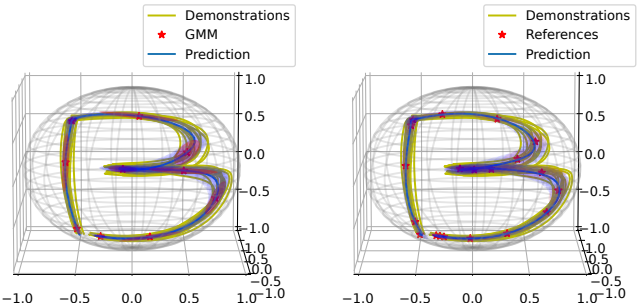
On $\mathcal{M}_2$ and $\mathcal{M}_3$, translating the origin would invalidate covariances, as angular DoFs, radius and identity orientation depend on the coordinate origin. We can however still exploit the properties of one model expressed in $\mathcal{M}_2$ or $\mathcal{M}_3$, fusing it with one or multiple models in $\mathcal{M}_1$. To this end, we use the coordinate origin of the model on $\mathcal{M}_2$ or $\mathcal{M}_3$ as global origin and move the other model outputs using (16) – (17) to this new origin, still expressing them on $\mathcal{M}_1$. Subsequently, using the coordinate transformation formulas and Jacobians given in [13], we transform means $\boldsymbol{\mu}_{\mathcal{M}_1}$ and covariances $\boldsymbol{\Sigma}_{\mathcal{M}_1}$ to $\boldsymbol{\mu}_{\mathcal{M}_{2,3}}$ and $\boldsymbol{\Sigma}_{\mathcal{M}_{2,3}}$. Finally, a Gaussian product on $\mathcal{M}_2$ or $\mathcal{M}_3$ fuses the result. This result can then be used to subsequently leverage properties of different $\mathcal{M}$, e.g., manifold-aware variable stiffness as presented in [13].

D. Evaluation in Toy Examples

To evaluate our Riemannian KMP prediction (Section III-A), we compare it to STS baselines [9] and a Riemannian GMR [4].[2] We encode a circular trajectory with $\boldsymbol{s} \in [0, 1]$ and an orientation change of 2π depicted as the uppermost line in Fig. 3. We sample 1000 poses of this trajectory modeled from 100 Gaussians centered at the nominal trajectory. After fitting a GMM with 5 Gaussians, we extract a probabilistic reference composed of 50 poses using GMR on equidistant time inputs $\boldsymbol{s}^* \in [0, 1]$ to fit the KMP. We continue using this approach of extracting a probabilistic reference throughout the experimental evaluation. For the STS approach, we prepare data points using $\boldsymbol{\xi}\left(\boldsymbol{s}\right)_{\mathrm{sts}} = \mathrm{Log}_{\boldsymbol{e}}\left(\boldsymbol{\xi}\left(\boldsymbol{s}\right)_{\mathcal{M}_1}\right)$ and transform the prediction using $\boldsymbol{\xi}\left(\boldsymbol{s}^*\right)_{\mathcal{M}_1} = \mathrm{Exp}_{\boldsymbol{e}}\left(\boldsymbol{\xi}\left(\boldsymbol{s}^*\right)_{\mathrm{sts}}\right)$. For our Riemannian approach, we employ (13). Both use a Radial Basis Function (RBF) kernel with $l = 0.1$. The results are visualized in Fig. 3 – while our approach is able to predict all orientations correctly, the STS approach fails completely for some orientations. As additional baseline, we use a Riemannian Nadaraya-Watson regression [28] on all sampled poses and a RBF kernel with $l = 0.007$. We omit plotting the Nadaraya-Watson result as it is visually indistinguishable from the KMP result. Table I summarizes positional and rotational error for all approaches averaged over 200 query points - while all models achieve a good accuracy for position prediction, the orientation prediction of the STS KMP [9] suffers from discontinuities of the log map. We achieve an accuracy comparable to that of the Nadaraya-Watson baseline, even though it has access to all data points. Note that this baseline cannot model epistemic uncertainty which is a key requirement e.g., in state-based policies; furthermore it has an increased runtime with many sample points as seen in Section V-D.

Figure 3 furthermore shows the adaptation using a via point above the original trajectory, including an orientation change, and a similar null space action (Section III-B).

Another evaluation of our proposed Riemannian KMP on $\mathcal{S}^2$ is depicted in Fig. 4, showcasing that the method works on different Riemannian manifolds and is not constrained to manifolds exhibiting a Lie group structure. Note that on $\mathcal{S}^2$, where unlike for $\mathcal{S}^1$ and $\mathrm{SO}(3)$, we cannot construct a singularity-free log map, there is at least one discontinuity. For our parameterization, this discontinuity is located at the south pole; models passing over it thus suffer from singularities similar to those seen for STS models in Table I. For trajectories only passing over parts of the sphere, valid models can be obtained by first recentering data around the north pole.



(a) Gaussian Mixture Regression. (b) Kernelized Movement Primitive.

Fig. 4: Plots for the letter "B" of the handwriting dataset [10] projected to $\mathcal{S}^2$. We fit the GMM using 10 Gaussians and use 20 reference points for KMP; all these Gaussians are depicted with their mean and covariance ellipsoid.

IV. MATRIX-VALUED KERNELS

Commonly employed kernels for vector-valued output, such as separable kernels (Section IV-A) used in *vanilla* KMP [8],

[1]On $\mathcal{S}^2$, parallel transport is required due to the hairy ball theorem.

[2]While including Riemannian GMR [4] in Table I, we do not seek to outperform it in general as the probabilistic reference is extracted from GMR, but to achieve comparable performance in regions of *known* data.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

TABLE II: MAXIMUM LENGTH SCALES FOR SELECTED KERNELS.

Kernel	$\mathcal{S}^1$	$\mathcal{S}^1 \times \mathbb{R}^2$	$\mathcal{S}^2$	SO(3)	$\mathbb{R}^3 \times$ SO(3)
RBF	0.01	0.65	0.27	0.63	1.1
Matérn ($\nu = \frac{3}{2}$)	0.40	0.99	0.77	0.97	1.46
Matérn ($\nu = \frac{5}{2}$)	0.07	0.83	0.58	0.8	1.33

do not measure per-DoF uncertainties. Instead, the output of a kernel with scalar output is assigned to all DoFs. To address this shortcoming, matrix-valued kernels measuring per-DoF uncertainties compatible with our proposed Riemannian formulation are required. In Section IV-B, we introduce a novel diagonal kernel computing this required covariance. We show the resulting differences on toy examples in Section IV-C.

A. Separable Kernels

A common choice for vector-valued output is the product of a kernel with scalar output and the identity matrix [8]

$$\boldsymbol{k}(\boldsymbol{s}, \boldsymbol{s}') = k(\boldsymbol{s}, \boldsymbol{s}')\boldsymbol{I}. \tag{18}$$

This type of kernel is called *separable* as the interactions between outputs are modeled without reference to the input space [19]. As such, these kernels can only output a scalar-valued epistemic uncertainty which is the same for all DoFs.

The resulting kernel matrix has only entries on its diagonal which are all identical. Parallel transporting results in

$$\boldsymbol{k}_{\parallel}(\boldsymbol{s}, \boldsymbol{s}') = \boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}} k(\boldsymbol{s}, \boldsymbol{s}') \boldsymbol{I} \boldsymbol{R}_{\boldsymbol{e}\to\boldsymbol{\mu}}^{\top} = k(\boldsymbol{s}, \boldsymbol{s}') \cdot \boldsymbol{I} \tag{19}$$

and thus does not lead to changes, irrespective of the chosen tangent space. Using the 6×6 matrix $\boldsymbol{I}_6$, this formulation can, e.g., be used to predict full pose outputs.

To use manifold-valued inputs in the RBF and Matérn kernels, we use the scaled Riemannian distance $d^{\mathcal{M}}(\boldsymbol{s}, \boldsymbol{s}')/l = \left|\left|\mathrm{Log}_{\boldsymbol{s}}^{\mathcal{M}}(\boldsymbol{s}')\right|\right|_2 / l = r$ to compute

$$k_{\mathrm{RBF}}^{\mathcal{M}}(r) = \exp\left(-\frac{1}{2}r^2\right) \tag{20}$$

$$k_{\mathrm{Matérn}}^{\mathcal{M}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} r\right)^{\nu} K_{\nu}\left(\sqrt{2\nu} r\right) \tag{21}$$

where l is the length scale parameter, ν a parameter of the Matérn kernel and K_ν a modified Bessel function [6]. As found by Feragen et al. [29], this distance metric does not result in positive definite RBF kernels for all choices of length scales. As later found by the same authors [30], for small enough length scales, positive definite kernels can be achieved which has been used by Jaquier et al. [31] to sample valid length scales.[3] We determine valid length scales using this approach for our kernels and summarize them in Table II. Separable kernels are used when *input* and *output dimensions* do not match – i.e., for time-based policies and task-parameterized models evaluated in Sections VI-A, VI-B, VI-D and VI-E. They can also be used for state-based policies where both dimensions agree, at the cost of only having a scalar-valued epistemic uncertainty as seen in Section VI-C.

B. Diagonal Kernels

To model DoF-aware kernels, we first propose to multiply a kernel with scalar output (20) – (21) with a diagonal kernel

$$\boldsymbol{k}(\boldsymbol{s}, \boldsymbol{s}') = k(\boldsymbol{s}, \boldsymbol{s}') \cdot \boldsymbol{K}_{\mathrm{diag}}(\boldsymbol{s}, \boldsymbol{s}'). \tag{22}$$

$\boldsymbol{K}_{\mathrm{diag}}$ is a diagonal matrix with the p-th diagonal entry

$$\boldsymbol{K}_{\mathrm{diag},p,p}(\boldsymbol{s}, \boldsymbol{s}') = k\left(\left|\left(\mathrm{Log}_{\boldsymbol{s}}^{\mathcal{M}}(\boldsymbol{s}')\right)_p\right|\right). \tag{23}$$

[3]At a computational and complexity cost, more involved kernels could ensure positive definiteness for all length scales [32].

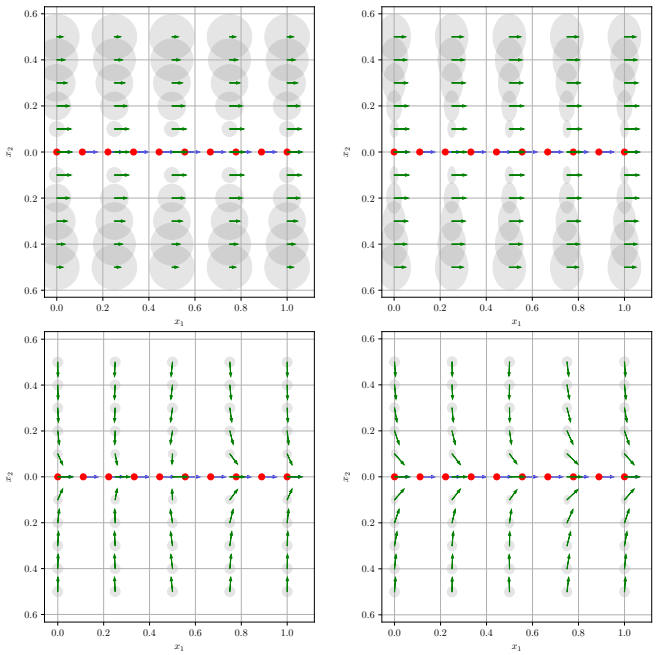


Fig. 5: Upper row: KMP output of a velocity policy with separable RBF kernel ($l = 0.3$ (left)) / diagonal RBF kernel ($l_{RBF} = 0.6$, $l_{\mathrm{diag}} = 0.3$ (right)). Lower row: fusion of the KMP with a *stabilizing policy* [13]. Red dots visualize covariances and locations of known velocities (blue); grey ellipses the covariance and green arrows the velocity of KMP and fusion, respectively.

Its diagonal elements are calculated from a kernel k (20)-(21) with input on $\mathbb{R}$ while all other entries are zero. Positive definiteness of the resulting kernel is ensured through its construction using the Hadamard product $\odot$ of a transformation kernel [33] and the identity matrix $\boldsymbol{I}$ and the Kronecker product with a kernel $k(\boldsymbol{s}, \boldsymbol{s}')$ with scalar output [34].

Note that this kernel is already, through the manifold logarithm $\mathrm{Log}_{\boldsymbol{s}}(\cdot)$ in (23), expressed in $\mathcal{T}_{\boldsymbol{s}}\mathcal{M}$, thus not requiring parallel transport. As before, valid length scales on different manifolds can be determined through sampling. Such kernels can be used when *input* and *output dimensions* agree. In Section VI-C, this degree-wise uncertainty is used to push the policy back into its valid region while keeping a forward motion component near regions of known data.

Among the manifolds considered in this work, we can formulate a valid kernel on $\mathbb{R}^n$, $\mathcal{S}^1$, $\mathcal{S}^3$ as well as their products, where $\mathrm{Log}_{\boldsymbol{s}}^{\mathcal{M}} \boldsymbol{s}' = -\mathrm{Log}_{\boldsymbol{s}'}^{\mathcal{M}} \boldsymbol{s}$ holds for our logarithm maps. On $\mathcal{S}^2$ as *input*, this property is not given due to the hairy ball theorem which topologically prevents us from creating a global, continuous frame. The result would be an asymmetric, and thus invalid, kernel which cannot be the result of kernelization applied to obtain the solution (13). On $\mathcal{S}^2$ as input, instead, a separable kernel only providing a scalar estimate of the epistemic uncertainty must be used.

C. Evaluation in Toy Examples

To show the usefulness of matrix-valued kernels, we learn velocity policies in 2D. As depicted in Fig. 5, points on $\boldsymbol{s} = \boldsymbol{x} \in \mathbb{R}^2$ (red) with known velocities on $\boldsymbol{\xi}(\boldsymbol{s}) = \dot{\boldsymbol{x}} \in \mathbb{R}^2$ (blue) are encoded in a KMP using different kernels. For the commonly used separable RBF kernel (Section IV-A), we only obtain scalar covariances; the kernel thus cannot output individual uncertainties per DoF. Conversely, per-DoF uncertainties can be calculated by the diagonal kernel (Section IV-B). The covariance along the x_2 axis gets larger than along the x_1 axis when moving away from known points along that axis, highlighting the higher uncertainty for that DoF.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

Algorithm 1 Prediction in a cKMP.

$\boldsymbol{L}\boldsymbol{L}^\top \leftarrow \boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ ▷ Initial fitting (24)
$\boldsymbol{\alpha} \leftarrow \boldsymbol{L}^\top \backslash \boldsymbol{L} \backslash \mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e})$ ▷ Pre-solve with (25)
for $\boldsymbol{s}_i^*$ in queries $[\boldsymbol{s}_0^* \dots \boldsymbol{s}_n^*]$ **do**
  $\boldsymbol{\xi}(\boldsymbol{s}_i^*) \leftarrow \mathrm{Exp}_{\boldsymbol{e}}(-\boldsymbol{k}^*\boldsymbol{\alpha})$ ▷ Predict mean (13)
  $\boldsymbol{V} \leftarrow \boldsymbol{L} \backslash \boldsymbol{k}^{*\top}$
  $\boldsymbol{\Sigma}(\boldsymbol{s}_i^*) \leftarrow \boldsymbol{k}(\boldsymbol{s}_i^*, \boldsymbol{s}_i^*) - \boldsymbol{V}^\top\boldsymbol{V}$ ▷ Predict covariance (14)
end for

Following [13], those learned policies are fused with a *stabilizing policy* with constant, diagonal covariance $\boldsymbol{\Sigma}_{\mathrm{stab}}$ and normalized velocity $\boldsymbol{\mu}_{\mathrm{stab}}$ pointing towards the closest point with known velocity. As seen in Fig. 5, for matrix-valued kernels (right side), the covariance first increases along the perturbed DoF while in the separable case (left side) the covariance increases for all DoFs. In the fused velocity output (lower row), the policy with diagonal kernel keeps a forward motion component while the policy with separable kernel focuses on returning to the region of known data. Depending on the closest point, this may even result in backwards motion as seen by the arrows for the fusion with separable kernel (lower left side) in the middle pointing slightly to the left while the fusion with diagonal kernel (lower right side) always has a forward motion component to the right side.

This result is even more pronounced with more DoFs as used in Section VI-C – using such kernels allows the policy to proceed and only be corrected by the *stabilizing policy* along DoFs with perturbations, given they are not too large, thus overcoming a central limitation in [13].

V. Computational Improvements

Fast fitting to data and quick adaptations to novel situations are desirable properties of movement primitives. In KMPs, just as in GPs, calculating the inverse required for mean (13) and covariance (14) prediction is the computationally most expensive operation and additionally prone to numerical instabilities. The Cholesky decomposition, as commonly used in GPs [6, 35], avoids calculating explicit inverses and improves execution speed in KMP predictions (Section V-A) as well as in adding (Section V-B) and removing (Section V-C) measurements. We adapt this approach to fitting, adding and removing points in our manifold-aware formulation and introduce a novel mechanism to restore the decomposition when removing points from the reference $\mathcal{D}$.

A. Cholesky-factorized KMP (cKMP)

Instead of calculating the inverse $(\boldsymbol{K} + \lambda\boldsymbol{\Sigma})^{-1}$, we use the Cholesky factorization of the positive definite matrix $\boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ into a product of lower triangular matrix $\boldsymbol{L}$ and its transpose

$$\boldsymbol{L}\boldsymbol{L}^\top = \boldsymbol{K} + \lambda\boldsymbol{\Sigma} \tag{24}$$

which allows for solving the equation $\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{x} = \boldsymbol{b}$ through a forward and a backward substitution to obtain $\boldsymbol{x}$

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{L}^\top\boldsymbol{x} = \boldsymbol{y}. \tag{25}$$

Algorithm 1 summarizes the mean and covariance prediction using Cholesky factorization, where we use $\backslash$ to denote the solving operation (25) using forward- or backward substitution, resulting in a similar algorithm to that of predictions in a Cholesky-factorized GP [6], however adapted to observation data on manifolds through the use of $\mathrm{Log}(\cdot)$ and $\mathrm{Exp}(\cdot)$. The computational complexity of the mean prediction is thus $\mathcal{O}\left(NM^2\right)$, where N is the size of $\boldsymbol{L}$ and M the output dimension, and $\mathcal{O}\left(N^2\right)$ for the covariance prediction dominated by the calculation of $\boldsymbol{V}$. While $\boldsymbol{\alpha}$ can also be precomputed for the vanilla KMP and iKMP prediction (13), the default covariance prediction (14) is $\mathcal{O}\left(2N^2M\right)$. This results in significantly improved prediction times of our Cholesky-factorized KMP (cKMP) compared to *vanilla* KMP [8].

Algorithm 2 Adding measurements to a cKMP.

for $\boldsymbol{s}_i$, $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ in *new measurements* **do**
  $\boldsymbol{s} \leftarrow [\boldsymbol{s}; \boldsymbol{s}_i]$, $\boldsymbol{\mu} \leftarrow [\boldsymbol{\mu}; \boldsymbol{\mu}_i]$, $\boldsymbol{\Sigma} \leftarrow [\boldsymbol{\Sigma}; \boldsymbol{\Sigma}_i]$ ▷ append
end for
$\boldsymbol{K} \leftarrow [\boldsymbol{K}; \boldsymbol{K}_{i,j}]$ ▷ Calculate & append new entries
$\boldsymbol{L} \leftarrow \mathrm{cholupdate}(\boldsymbol{K} + \lambda\boldsymbol{\Sigma})$ ▷ through (26) – (28)

TABLE III: Computation times (milliseconds) for adding and removing 3 points and predicting 200 points for letter writing.

Method	Fit (V-A)	Add (V-B)	Remove (V-C)	Predict (V-A)
N.-W. [18]	0.12 ± 0.00	-/-	-/-	0.83 ± 0.72
KMP [8]	12.93 ± 3.93	13.63 ± 3.97	11.42 ± 3.16	1.68 ± 0.33
iKMP [21]	12.17 ± 3.01	1.18 ± 0.09	0.84 ± 2.33	1.71 ± 0.58
cKMP (V)	$\mathbf{2.66 \pm 0.78}$	$\mathbf{0.82 \pm 0.16}$	$\mathbf{0.60 \pm 0.58}$	$\mathbf{1.35 \pm 0.59}$

B. Appending Measurements

Osborne [36, B.1] proposes an algorithm for adding rows and columns to a Cholesky factorization which we can readily use to calculate an updated $\boldsymbol{L}^*$ when appending new entries to the end of the matrix $\boldsymbol{A} = \boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ we want to factorize. Setting the original $\boldsymbol{A}$ as the upper left corner of $\boldsymbol{A}^*$

$$\boldsymbol{A}^* = \begin{pmatrix} \boldsymbol{A}_{1,1}^* & \boldsymbol{A}_{1,2}^* \\ \boldsymbol{A}_{2,1}^* & \boldsymbol{A}_{2,2}^* \end{pmatrix}, \quad \boldsymbol{L}^* = \begin{pmatrix} \boldsymbol{L}_{1,1}^* & \boldsymbol{0} \\ \boldsymbol{L}_{2,1}^* & \boldsymbol{L}_{2,2}^* \end{pmatrix}, \tag{26}$$

where $\boldsymbol{A} = \boldsymbol{A}_{1,1}^*$, and through $\boldsymbol{A}^* = \boldsymbol{L}^*\boldsymbol{L}^{*\top}$, we obtain

$$\boldsymbol{L}_{1,1}^* = \boldsymbol{L}, \quad \boldsymbol{L}_{1,1}^*\boldsymbol{L}_{2,1}^{*\top} = \boldsymbol{A}_{1,2}^*, \tag{27}$$

$$\boldsymbol{L}_{2,2}^*\boldsymbol{L}_{2,2}^{*\top} = \boldsymbol{A}_{2,2}^* - \boldsymbol{L}_{2,1}^*\boldsymbol{L}_{2,1}^{*\top}. \tag{28}$$

We can thus reuse the precomputed $\boldsymbol{L}$ as upper left block $\boldsymbol{L}_{1,1}^*$, $\boldsymbol{L}_{2,1}^*$ can then be obtained using forward substitution and $\boldsymbol{L}_{2,2}^*$ through a Cholesky decomposition.

The full algorithm is denoted in Algorithm 2. As we add measurements to the end of the decomposition, the complexity of (27) is $\mathcal{O}(N^2)$ where N is the size of the original $\boldsymbol{L}$ and the complexity of (28) is $\mathcal{O}(D^3)$ where D is the dimension of the measurements to be added. This results in a clear computational advantage over the naive recomputation of the inverse used by the original KMP and is also significantly faster than iKMP (Tables III and IV).

C. Removing Measurements

For removing a measurement, we permute $\boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ with a permutation matrix $\boldsymbol{P}$ to obtain

$$\boldsymbol{P}\left(\boldsymbol{K} + \lambda\boldsymbol{\Sigma}\right)\boldsymbol{P}^\top = \boldsymbol{P}\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{P}^\top = \boldsymbol{L}'\boldsymbol{L}'^\top. \tag{29}$$

Note that in case we only want to remove measurements at the end (which is often required when we iteratively update a via point) $\boldsymbol{P} = \boldsymbol{I}$ and we can simply remove the last rows and columns of $\boldsymbol{L}$, $\boldsymbol{K}$ and $\boldsymbol{\Sigma}$ to obtain $\boldsymbol{L}'$, $\boldsymbol{K}'$ and $\boldsymbol{\Sigma}'$, resulting in a negligible computational cost. In case we want to remove a measurement before the end, the lower-triangular form of $\boldsymbol{L}'$ has to be restored as shown in Appendix B-B.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

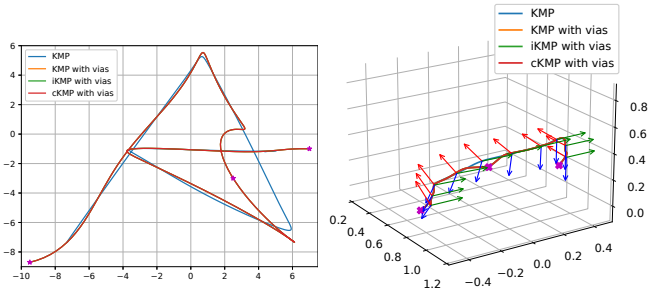


Fig. 6: Adapting the letter "A" in the letter writing task [10] (left) and a full trajectory of 500 poses on $\mathcal{M}_1$ (right; via points shown as magenta $\boldsymbol{x}$). Adapted plots for different calculation methods are indistinguishable as they result in the same trajectory, indicating their numerical equivalence.

TABLE IV: COMPUTATION TIMES (MILLISECONDS) FITTING, ADDING 3 POSES AND SUBSEQUENTLY REMOVING THEM IN A TRAJECTORY WITH 100 REFERENCE POSES, PREDICTING 500 POSES.

Method	Fit (V-A)	Add (V-B)	Remove (V-C)	Predict (V-A)
N.-W. [18]	0.22 ± 0.01	-/-	-/-	24.06 ± 2.89
R. KMP (III)	37.11 ± 3.74	37.20 ± 1.72	34.51 ± 2.61	13.28 ± 1.22
iKMP [21]	36.22 ± 3.35	4.11 ± 1.60	2.29 ± 0.52	13.15 ± 0.95
cKMP (V)	$\mathbf{7.89 \pm 2.32}$	$\mathbf{2.64 \pm 0.32}$	$\mathbf{1.27 \pm 0.37}$	$\mathbf{8.57 \pm 0.82}$

D. Evaluation in Toy Examples

We provide computation times for the original KMP formulation, iKMP [21] and the Cholesky-factorized KMP (cKMP) introduced in this section for initial fitting (*vanilla* KMP / iKMP: calculation of the inverses, cKMP: Cholesky decomposition, both: calculation of $\boldsymbol{\alpha}$ (Algorithm 1)), adding of 3 via points and subsequent removal of those as well as the full prediction for letter writing in Table III and for a full trajectory prediction on $\mathcal{M}_1$ (Section VI-D) in Table IV (both shown in Fig. 6).[4] As baseline, we compare with a Riemannian version of Nadaraya-Watson regression ([28]) inspired by [18] – note that while this baseline compares favorably on small problems, for movements with many points, it quickly gets much slower as it takes every single data point into account at prediction time. This method furthermore can only provide an *aleatoric* uncertainty estimate, but no *epistemic* uncertainty. Furthermore, via points can only be added by using custom engineered kernels as suggested by [18]; we have omitted their implementation. Adding nullspace modifications to the KMP prediction does not measurably increase computation times; timings are thus omitted from Tables III and IV.

Initial fitting, adding and removing points in the original KMP formulation take approximately the same time, as a full inversion of $\boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ is required. Even though we also precompute $\boldsymbol{\alpha}$ (Algorithm 1) for the *vanilla* KMP and iKMP, our proposed cKMP is still significantly faster due to a quicker covariance calculation. Note that for adding and removing points in the original KMP, already reallocating smaller or bigger memory areas for $\boldsymbol{K}$ and $\boldsymbol{\Sigma}$ takes considerable time as each matrix with $(D \cdot N)^2 = 600^2$ entries for $D = 6$, $N = 100$ in double precision occupies 2.88 MB of memory. In iKMP, only the inverse and in cKMP, only the Cholesky decomposition have to be updated while full $\boldsymbol{K}$ and $\boldsymbol{\Sigma}$ are only calculated for initial fitting.

VI. EVALUATION ON A ROBOTIC SYSTEM

We evaluate our approach in real experiments that either would not have been possible before or benefit significantly from our contributions. We use the on-manifold impedance and velocity controllers of [13]. All experiments use the Riemannian KMP prediction (Section III-A). The first two experiments (Sections VI-A and VI-B) furthermore employ task parameterization (Section III-C), the third experiment (Section VI-C) demonstrates the usefulness of matrix-valued kernels (Section IV) in DSs. In Section VI-D, we use the movement primitive adaptation (Section III-B). This experiment also profits exceptionally from the computational improvements (Section V), allowing for a continuous trajectory update. The final experiment in Section VI-E highlights the importance of an epistemic uncertainty estimate. We show the execution of all experiments in the supplementary video.

[4]Timings and their standard deviations were computed for 100 executions on a computer with Intel Core i7-10700K CPU using Ubuntu 24.04. The computations were performed and measured in parallelized C++ code compiled with AVX optimizations enabled.

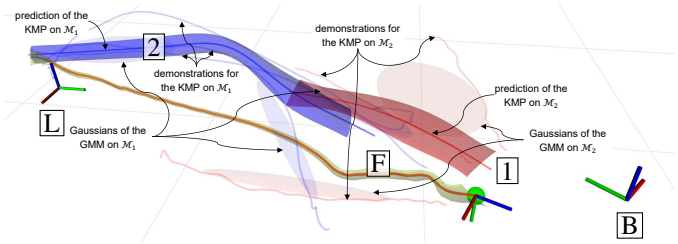


Fig. 7: Riemannian task-parameterized KMP on $\mathcal{M}_2$ for picking up a bottle from any angle at B and placing it at L. The model 1 on $\mathcal{M}_2$ (red demonstrations / Gaussians / covariance; origin B) is fused with a model 2 on $\mathcal{M}_1$ (blue demonstrations / Gaussians / covariance; origin L) to a model F on $\mathcal{M}_2$ (yellow covariance).

A. Task Parameterization on Different Manifolds

We first evaluate task parameterization on $\mathcal{M}_1$ and $\mathcal{M}_2$ (Section III-C) on the task of grasping a bottle B, which is a rotationally symmetric problem, and placing it at a specified location L (Fig. 7). The KMP 1 is learned from three demonstrations of picking up the bottle from different angles at B. By using $\mathcal{M}_2$ centered at B, this results in a high covariance along the angular DoF which we use to modulate a low stiffness. Note that with the control frame located at the end effector behind the gripper, the trajectory never visually "reaches" the bottle location B. The KMP 2 on $\mathcal{M}_1$ is learned from 3 demonstrations of placing the bottle at L in a relatively constrained manner, thus resulting in lower covariance and stronger guidance along the angular DoF when fused on $\mathcal{M}_2$ depicted as F. From this combination of policies, the operator can pick the bottle B from different angles, while the placement at L is constrained. The *aleatoric* uncertainty plays a key role in this experiment – both KMPs are queried in a region with available data and thus low *epistemic* uncertainty. The uncertainty associated with those known data points accordingly determines both the fused result and the stiffness. Note that the rotational part of the covariance as well as the couplings between position and orientation also influence the fusion result, neither of which can be visualized in Fig. 7. The supplementary video shows the resulting impedance behavior (stabilized using [37]) of the robot along the complete task.

B. Comparison to Vanilla Task-Parameterized KMPs

We compare our task-parameterized KMP on the measurement ring insertion task from [38, 39] using their Open-Source implementation[5] with both the vanilla task-parameterized KMP available in the code and our Riemannian implementation on $\mathcal{M}_1$. Fig. 8 shows the setup used for training and evaluation. We use 15 demonstrations in total, 10 for the $0°$ and 5 for $90°$ measurement station orientations shown

[5]https://github.com/DLR-RM/interactive-incremental-learning

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

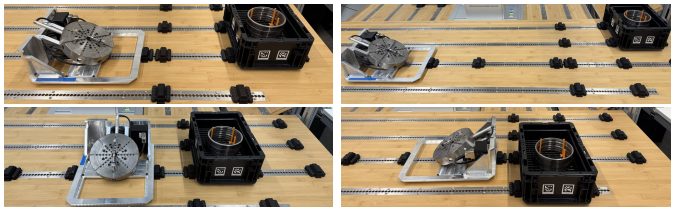


Fig. 8: Experimental setup for grasping bearing rings from a box and inserting them in a measurement station. The station is rotated at $0°$ (upper row) with two different offsets and at $90°$ and $180°$ (lower row). While rotations of $0°$ and $90°$ are used for training the task-parameterized model on $\mathcal{M}_1$, the rotation of $180°$ is only seen at test time.

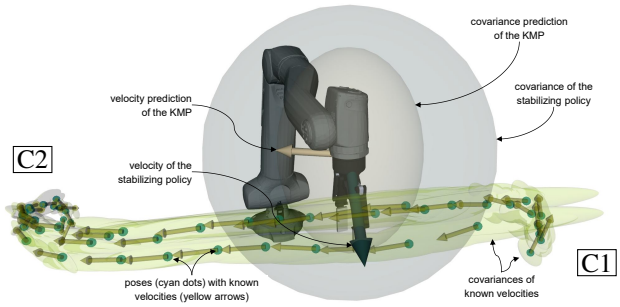


Fig. 9: Riemannian KMP with diagonal Matérn kernel for a state-based policy ($\boldsymbol{s} = \boldsymbol{x} \in \mathcal{M}_1$, $\boldsymbol{\xi}(\boldsymbol{s}) = \dot{\boldsymbol{x}} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}_1$) to move a ring from C1 to C2. Note the higher covariance of the KMP prediction around the perturbed z axis, leading to a correction by the *stabilizing policy* only along this DoF.

therein, corresponding to the possible locations of the black box. *Task frames* are recorded as the end effector location when closing the gripper for picking up the ring respectively when opening the gripper for releasing it. We train two KMPs, one for picking and one for placing the ring in the respective coordinate frames. At test time, these task frames are set to box and measurement station locations to allow for generalization.

As seen in the supplementary video, while the vanilla implementation succeeds when it is only trained with one orientation, as soon as samples with the measurement station rotated at $90°$ are added, the orientation prediction fails. This happens as treating quaternions as vector-valued data in the original KMP neglects their geometry, such that even after renormalization, the obtained orientation is meaningless. Contrarily, our approach succeeds in the insertion task for all possible combinations, providing smooth trajectories also for the case of the measurement station rotated by $180°$ which has not been part of the training data.

C. Dynamical Systems with Matrix-valued Kernels

To evaluate matrix-valued kernels (Section IV), we encode a velocity policy for moving a ring from C1 to C2 (Fig. 9) learned from four kinesthetic demonstrations in a KMP using both a separable Matérn kernel with $\nu = \frac{3}{2}$ and $l = (0.15, 0.15, 0.15, 0.2, 0.2, 0.2)$ and a diagonal Matérn kernel using the same parameters for the diagonal part and $l = (0.4, 0.4, 0.4, 0.8, 0.8, 0.8)$ for the scalar kernel. The

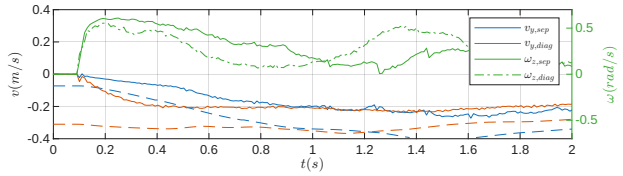


Fig. 10: Velocities in y (measured: solid; desired: dashed) and around the z axis using a separable (subscript *sep*) and a diagonal Matérn kernel (subscript *diag*) when starting with a rotational perturbation around the z axis at $t = 0$.

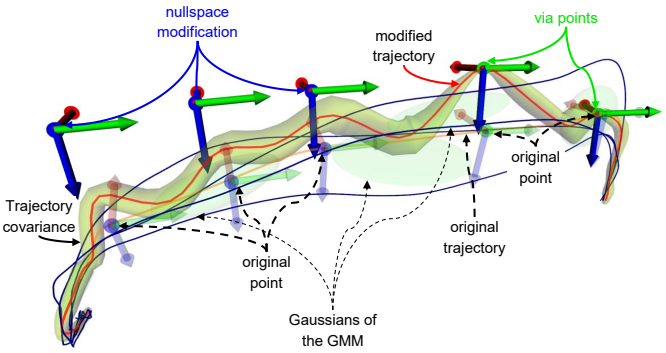


Fig. 11: Real-time adaptation of a trajectory with different modifications.

learned policy (Fig. 9) is fused with a *stabilizing policy* [13] pointing towards the closest known pose with velocity $\mu_{\text{stab}} = 0.5\,\text{m}\,\text{s}^{-1}$ and $\boldsymbol{\Sigma}_{\text{stab}} = 0.1\,\text{m}^2 \cdot \boldsymbol{I}$ [13, (26)].

While both kernels result in similar motions for the unperturbed case, their perturbed executions differ. We start the policy at $t = 0$ in the middle of the trajectory with a perturbation of $-0.7\,\text{rad}$ around the z axis. As seen in Fig. 10 and the supplementary video, while the *stabilizing policy* corrects the rotation around the z axis similarly, the system accelerates more quickly with the diagonal kernel along the y axis – the main motion direction. This is possible as the diagonal kernel can model per-DoF *epistemic* covariances during perturbations, thus allowing the policy to control the motion along y earlier than in the separable case.

Note that beyond these differences, the experiment in this section is enabled by the *epistemic uncertainty* estimate of our approach – e.g., GMMs which do only output *aleatoric uncertainty* cannot model distance to known data. The *stabilizing policy* would thus never get activated, leading to ever increasing errors in case of perturbations.

D. Adapting Kernelized Trajectory Fixtures

Next, we show the adaptation of a time-based trajectory ($\boldsymbol{s} \in \mathbb{R}$, $\boldsymbol{\xi}(\boldsymbol{s}) \in \mathcal{M}_1$) learned from 4 demonstrations using via points and null space adaptations (Section III-B) in the same experimental run of a shared control scenario. Both modification types are compatible and can be chosen according to task requirements. While we physically interact with the robot, a teleoperated coupling would also be possible. The task to be performed by a human operator guided by a trajectory (Fig. 11) is to pick a ring from a container and place it on the measurement station while avoiding an unmodeled obstacle. We use a Matérn kernel ($\nu = \frac{5}{2}$, $l = 0.3$) for the KMP, a covariance of $\boldsymbol{\Sigma} = 1 \times 10^{-8}\,\text{m}^2$ for via points and decorrelate null space actions using the approach introduced in [3] to achieve more intuitive modifications. This experiment is performed with the end effector rotated by $180°$ around the y-axis – the setting where the STS KMP fails in Section III-D.

The user can start a modification by clicking a button on the robot. Depending on the selected mode, either a new via point or a new null space modification with the current end effector pose is applied at the current $\boldsymbol{s}$ of the trajectory. This modification is continuously updated by removing the previous modification and applying a new one at the new end effector pose while preserving previous via points and null space actions. Using the computational improvements introduced in Section V, calculating the update and updating the trajectory in the controller for 100 extracted poses takes less than $10\,\text{ms}$ compared to $40\,\text{ms}$ without. We use a constant update rate of $10\,\text{Hz}$, while fading between subsequent trajectories in the controller over $\Delta t = 0.5\,\text{s}$ from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$ along the Riemannian geodesic. This update rate is chosen deliberately

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

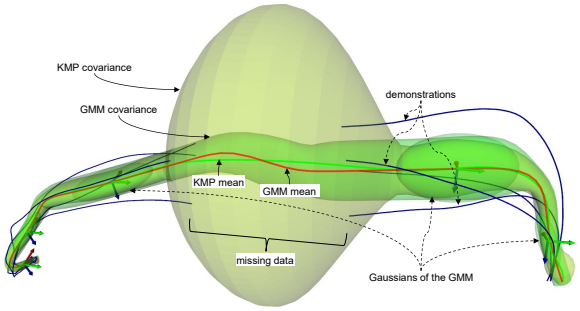


Fig. 12: Comparison of Riemannian GMM and KMP predictions with missing data. Missing demonstration data, signified by cut outs of the demonstration trajectories, leads to increased covariance of the KMP prediction while the GMM incorrectly predicts high certainty.

at just 2 Hz in order not to close the control loop through the adaptation, avoiding potential instabilities. A modification to circumvent an obstacle is shown in Fig. 11, where 2 via points and 3 null space modifications have been applied to correct both position and orientation. While via points achieve a *precise* modification, with the trajectory passing directly through them, they also lower the predicted uncertainty. Contrarily, null space modifications are more *coarse* and depend on and preserve the learned covariance – larger modifications are thus required in the lower covariance region on the left to achieve the same modification as in the middle.

E. Kernelized Trajectory Fixtures With Missing Data

To evaluate the significance of the epistemic uncertainty prediction of our approach, we compare with a GMM as baseline. As seen in Fig. 12, the GMM incorrectly predicts low uncertainty in regions of missing data while the epistemic uncertainty captured by our approach predicts an increased uncertainty. This covariance prediction can be used by downstream algorithms [13, 37] to reduce the stiffness of the robot in uncertain regions as shown in the supplementary video.

VII. Discussion

Our work leverages the expressiveness of Gaussian distributions on different Riemannian manifolds $\mathcal{M}$. By modeling the motion on a manifold including the orientation, as opposed to a purely Euclidean modeling, we can represent orientations and support the full range of possible rotations. Depending on the manifold, e.g., rotationally symmetric tasks can be learned and fused efficiently using our extended task parameterization.

We believe our method is best used in scenarios where explainability and fast computations are required, such as in shared control with humans in the loop. Especially applications with few available demonstrations, a need for fast adaptations and high precision requirements benefit from our approach. Ensuring manifold-valued and reasonable outputs, our method output can directly be fed to controllers without additional safety layers, as demonstrated in the robotic experiments.

We have successfully demonstrated operations with up to six dimensions; higher dimensional input like images would, however, require preprocessing such as task frame extraction. Finally, generalization of the model is limited to areas close to available data – further away, the prediction converges to the process mean $\boldsymbol{e}$, while the model outputs a high epistemic uncertainty that can, e.g., be used to activate a *stabilizing policy*.

VIII. Conclusion

We have presented a Riemannian approach to probabilistic non-parametric movement modeling, providing both aleatoric and epistemic uncertainties. The on-manifold formulation allows us to model motions with arbitrary orientation changes, without being limited to an area around a linearization point. Through an extended task parameterization scheme, the properties of different manifolds can be leveraged. We overcome the limitation of commonly used separable kernels which are only able to predict a scalar-valued epistemic uncertainty by proposing a novel diagonal kernel, which allows us to better model state-based policies. Finally, through computational improvements by using the Cholesky decomposition and by using two adaptation schemes, trajectory modification, e.g., through kinesthetic human input, is possible in real time. While both non-separable kernels and computational improvements could also be applied to the Euclidean KMP, they are even more important when learning additional DoFs. In future work, we plan to leverage additional matrix-valued kernels, e.g., curl-free kernels [19] which can also predict off-diagonal epistemic uncertainty and to showcase our approach on discrete geometries.

References

[1] M. Mühlbauer, M. Chalon, J. Silvério, and A. Albu-Schäffer, "Control system for a torque-sensitive space robotic arm," in *76th International Astronautical Congress, IAC 2025*, ser. IAC 2025. International Astronautical Federation (IAF), 2025, pp. 310–321, doi: 10.52202/083088-0033.

[2] A. Schmidt, D. Höhn, H. Singh, M. Walter, M. Mühlbauer, E.-U. Pap, M. Panzirsch, B. Lodewikus, H. Van Strien, S. Zhang, A. Dammann, F. Broghammer, A. Hahn, T. Noack, N. Y. Lii, A. Albu-Schäffer, and T. Hulin, "Conceptual framework for teleoperated robotic recovery of large subsea UXOs," in *Conference on Telepresence*. IEEE, 2025, pp. 197–202, doi: 10.1109/telepresence66096.2025.11521536.

[3] G. Quere, F. Stulp, D. Filliat, and J. Silvério, "A probabilistic approach for learning and adapting shared control skills with the human in the loop," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 05 2024, pp. 15 728–15 734, doi: 10.1109/icra57147.2024.10610956.

[4] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1240–1247, 2017, doi: 10.1109/LRA.2017.2657001.

[5] S. Calinon, "Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control," *IEEE Robotics & Automation Magazine*, vol. 27, no. 2, pp. 33–45, 2020, doi: 10.1109/mra.2020.2980548.

[6] C. K. I. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006.

[7] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021, doi: 10.1007/s10994-021-05946-3.

[8] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019, doi: 10.1177/0278364919846363.

[9] Y. Huang, F. J. Abu-Dakka, J. Silvério, and D. G. Caldwell, "Toward orientation learning and adaptation in cartesian space," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 82–98, 2021, doi: 10.1109/tro.2020.3010633.

[10] J. Silvério and Y. Huang, "A non-parametric skill representation with soft null space projectors for fast generalization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2988–2994, doi: 10.1109/ICRA48891.2023.10161065.

[11] J. Silvério, Y. Huang, F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Uncertainty-aware imitation learning using kernelized movement primitives," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 90–97, doi: 10.1109/iros40897.2019.8967996.

[12] M. Knauer, A. Albu-Schäffer, F. Stulp, and J. Silvério, "Interactive incremental learning of generalizable skills with local trajectory modulation," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3398–3405, 2025, doi: 10.1109/lra.2025.3542209.

[13] M. Mühlbauer, B. Weber, S. Calinon, F. Stulp, A. Albu-Schäffer, and J. Silvério, "A unified framework for probabilistic dynamic-, trajectory- and vision-based virtual fixtures," *IEEE Transactions on Robotics*, 2026, doi: 10.1109/TRO.2026.3751889.

[14] M. Mühlbauer, T. Hulin, B. Weber, S. Calinon, F. Stulp, A. Albu-Schäffer, and J. Silvério, "A probabilistic approach to multi-modal adaptive virtual fixtures," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5298–5305, 2024, doi: 10.1109/LRA.2024.3384759.

[15] A. Padalkar, F. Stulp, G. Neumann, and J. Silvério, "Towards safe and efficient learning in the wild: Guiding RL with constrained uncertainty-aware movement primitives," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 6880–6887, 2025, doi: 10.1109/lra.2025.3566599.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

[16] N. Jaquier, L. Rozo, and T. Asfour, "Unraveling the single tangent space fallacy: An analysis and clarification for applying Riemannian geometry in robot learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 242–249, doi: 10.1109/icra57147.2024.10611701.

[17] G. Li, P. Xu, R. Wang, Q. Ye, J. Chen, D. Song, and Y. Huang, "Orientation learning and adaptation towards simultaneous incorporation of multiple local constraints," 2025. [Online]. Available: https://arxiv.org/abs/2510.07986

[18] P. C. Lopez-Custodio, K. Bharath, A. Kucukyilmaz, and S. P. Preston, "Non-parametric regression for robot learning on manifolds," 2024. [Online]. Available: https://arxiv.org/abs/2310.19561

[19] M. A. Álvarez, L. Rosasco, and N. D. Lawrence, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012, doi: 10.1561/2200000036.

[20] M. Micheli and J. A. Glaunès, "Matrix-valued kernels for shape deformation analysis," 2013. [Online]. Available: https://arxiv.org/abs/1308.5739

[21] Y. Wang, S. Wu, C. Liu, C. Zhang, J. Silvério, and Y. Huang, "A computationally efficient nonparametric approach for robot imitation learning," in *Proceedings of 2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[22] B. Ti, A. Razmjoo, Y. Gao, J. Zhao, and S. Calinon, "A geometric optimal control approach for imitation and generalization of manipulation skills," *Robotics and Autonomous Systems*, vol. 164, p. 104413, 2023, doi: 10.1016/j.robot.2023.104413.

[23] P. Thomas Fletcher, "Geodesic regression and the theory of least squares on Riemannian manifolds," *International Journal of Computer Vision*, vol. 105, no. 2, pp. 171–185, 2013, doi: 10.1007/s11263-012-0591-y.

[24] H. J. Kim, N. Adluru, M. D. Collins, M. K. Chung, B. B. Bendlin, S. C. Johnson, R. J. Davidson, and V. Singh, "Multivariate general linear models (MGLM) on Riemannian manifolds with applications to statistical analysis of diffusion weighted images," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 2705–2712, doi: 10.1109/cvpr.2014.352.

[25] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, Sep 2015, doi: 10.1007/s11370-015-0187-9.

[26] A. Mallasto and A. Feragen, "Wrapped Gaussian process regression on Riemannian manifolds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[28] M. Mühlbauer, A. Sachtler, A. O. Albu-Schäffer, and J. Silvério, "Geometric and model priors in motion primitives," in *2026 IEEE International Conference on Robotics and Automation, Workshop on Geometry in the Age of Data-Driven Robotics*, 2026.

[29] A. Feragen, F. Lauze, and S. Hauberg, "Geodesic exponential kernels: When curvature and linearity conflict," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3032–3042, doi: 10.1109/cvpr.2015.7298922.

[30] A. Feragen and S. Hauberg, "Open problem: Kernel methods on manifolds and metric spaces. what is the probability of a positive definite geodesic exponential kernel?" in *29th Annual Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, vol. 49. Columbia University, New York, New York, USA: PMLR, 23–26 Jun 2016, pp. 1647–1650. [Online]. Available: https://proceedings.mlr.press/v49/feragen16.html

[31] N. Jaquier, L. Rozo, S. Calinon, and M. Bürger, "Bayesian optimization meets Riemannian manifolds in robot learning," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 233–246. [Online]. Available: https://proceedings.mlr.press/v100/jaquier20a.html

[32] N. Jaquier, V. Borovitskiy, A. Smolensky, A. Terenin, T. Asfour, and L. Rozo, "Geometry-aware bayesian optimization in robotics using Riemannian Matérn kernels," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 164. PMLR, 08–11 Nov 2022, pp. 794–805.

[33] A. Caponnetto, C. A. Micchelli, M. Pontil, and Y. Ying, "Universal multi-task kernels," *The Journal of Machine Learning Research*, vol. 9, pp. 1615–1646, 2008.

[34] C. Carmeli, E. de Vito, A. Toigo, and V. Umanitá, "Vector valued reproducing kernel hilbert spaces and universality," *Analysis and Applications*, vol. 08, no. 01, pp. 19–61, 2010, doi: 10.1142/s0219530510001503.

[35] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[36] M. A. Osborne, "Bayesian Gaussian processes for sequential prediction, optimisation and quadrature," Ph.D. dissertation, Oxford University, UK, 2010.

[37] M. Mühlbauer, N. Werner, R. Balachandran, T. Hulin, J. Silvério, F. Stulp, and A. Albu-Schäffer, "Passive variable impedance for shared control," 2026, doi: 10.48550/ARXIV.2604.20557. [Online]. Available: https://arxiv.org/abs/2604.20557

[38] M. Knauer, S. Bustamante, T. Eiband, A. Albu-Schaffer, F. Stulp, and J. Silverio, "Irosa: Interactive robot skill adaptation using natural language," *IEEE Robotics and Automation Letters*, vol. 11, no. 4, pp. 5246–5253, 2026, doi: 10.1109/lra.2026.3671560.

[39] M. Knauer, V. Gieraths, T. Mai, S. Bustamante, A. Albu-Schäffer, F. Stulp, and J. Silvério, "Clasp: Language-driven robot skill selection and composition using task-parameterized learning," 2026, doi: 10.48550/ARXIV.2606.08169. [Online]. Available: https://arxiv.org/abs/2606.08169

[40] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014, doi: 10.1109/TRO.2014.2298059.

[41] R. Murray-Smith and A. Girard, "Gaussian process priors with arma noise models," in *Irish Signals and Systems Conference*, 2001.

[42] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

APPENDIX A
FUNCTION-SPACE DERIVATION OF RIEMANNIAN KMPS

Alternatively to the weight-space derivation of Riemannian KMPs in Section III-A, we can also derive them, similarly to GPs, in function space to show the similarities to Riemannian Gaussian Conditioning. To this end, we first revisit Gaussian Conditioning [4] in Appendix A-A using an on-manifold optimization scheme. We then show that Riemannian KMPs can be formulated similarly, leading to the same result as before in Appendix A-B.

A. On-Manifold Gaussian Conditioning

A Riemannian Gaussian Distribution with $\boldsymbol{\zeta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be decomposed into

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{\xi}(\boldsymbol{s}) \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^I \\ \boldsymbol{\mu}^O \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}^{II} & \boldsymbol{\Sigma}^{IO} \\ \boldsymbol{\Sigma}^{OI} & \boldsymbol{\Sigma}^{OO} \end{bmatrix}, \quad (30)$$

to compute the probabilistic output $\boldsymbol{\xi}(\boldsymbol{s})$ given an input $\boldsymbol{s}$ through Gaussian conditioning. $\boldsymbol{\xi}(\boldsymbol{s})$ is computed through the minimization problem (see, e.g., [27] for the Euclidean case)

$$\begin{aligned} J(\boldsymbol{\xi}(\boldsymbol{s})) &= \frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{\zeta})^\top \boldsymbol{\Sigma}^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{\zeta}) && (31) \\ &= c + \mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s}))^\top \boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}^I}(\boldsymbol{s}) + \\ &\quad \frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s}))^\top \boldsymbol{\Lambda}^{OO}\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) && (32) \end{aligned}$$

where c is a constant unaffected by $\boldsymbol{\xi}(\boldsymbol{s})$ and we have made use of the partitioning of the inverse of the covariance matrix

$$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}^{II} & \boldsymbol{\Lambda}^{IO} \\ \boldsymbol{\Lambda}^{OI} & \boldsymbol{\Lambda}^{OO} \end{bmatrix}. \quad (33)$$

A Taylor expansion of the minimization problem (32) yields

$$\begin{aligned} J(\boldsymbol{\xi}(\boldsymbol{s})) \approx c + \left(\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) + \Gamma_O(\boldsymbol{\epsilon})\right)^\top \boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}^I}(\boldsymbol{s}) + \\ \frac{1}{2}\left(\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) + \Gamma_O(\boldsymbol{\epsilon})\right)^\top \boldsymbol{\Lambda}^{OO}\left(\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) + \Gamma_O(\boldsymbol{\epsilon})\right) \end{aligned} \quad (34)$$

where $\Gamma_O(\boldsymbol{\epsilon})$ denotes the parallel transport $\Gamma_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}(\boldsymbol{\epsilon})$, thus transporting the update $\boldsymbol{\epsilon}$ we want to apply to the current estimate of $\boldsymbol{\xi}(\boldsymbol{s})$ from the tangent space $\mathcal{T}_{\boldsymbol{\xi}(\boldsymbol{s})}\mathcal{M}$ to $\mathcal{T}_{\boldsymbol{\mu}^O}\mathcal{M}$. Using parallel transport avoids the construction of an explicit adjoint operation on the manifold [24], also note the similarity in formulation to the Lie group approach [40] for fusing Gaussians. For the manifolds in this work, $\Gamma_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}(\boldsymbol{\epsilon})$ can be formulated as left multiplication of a rotation matrix $\boldsymbol{R}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\epsilon}$ which allows calculating the derivative of (34) w.r.t. $\boldsymbol{\epsilon}$. In a Gauss-Newton optimization scheme, the next update is found by setting the derivative to zero, resulting in

$$\begin{aligned} \boldsymbol{R}^\top_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\Lambda}^{OO}\boldsymbol{R}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\epsilon} = -\boldsymbol{R}^\top_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\Lambda}^{OO}\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) \\ -\boldsymbol{R}^\top_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}^I}(\boldsymbol{s}). \end{aligned} \quad (35)$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

Using the identity for the inverse of a partitioned matrix [35], it can be further simplified to

$$\begin{aligned}\boldsymbol{\epsilon} = &\quad \boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) \quad \boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\left(\boldsymbol{\Lambda}^{OO}\right)^{-1}\boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}^I}(\boldsymbol{s}) \\ = &\quad \boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\mathrm{Log}_{\boldsymbol{\mu}^O}(\boldsymbol{\xi}(\boldsymbol{s})) + \boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\Sigma}^{OI}\left(\boldsymbol{\Sigma}^{II}\right)^{-1}\mathrm{Log}_{\boldsymbol{\mu}^I}(\boldsymbol{s})\end{aligned} \tag{36}$$

and is in line with the results of [4]. Note that as the parallel transport is a rotation in our case, $\boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O} = \boldsymbol{R}^{-1}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}$ holds. $\boldsymbol{\xi}(\boldsymbol{s})$ can then be computed iteratively through $\boldsymbol{\xi}(\boldsymbol{s}) \leftarrow \mathrm{Exp}_{\boldsymbol{\xi}(\boldsymbol{s})}(\boldsymbol{\epsilon})$, after convergence, the covariance is computed to

$$\begin{aligned}\boldsymbol{\Sigma} &= \left(\boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\boldsymbol{\Lambda}^{OO}\boldsymbol{R}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\right)^{-1} \\ &= \boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}\left(\boldsymbol{\Sigma}^{OO} - \boldsymbol{\Sigma}^{OI}\left(\boldsymbol{\Sigma}^{II}\right)^{-1}\boldsymbol{\Sigma}^{IO}\right)\boldsymbol{R}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O},\end{aligned} \tag{37}$$

again using the formulas for the inverse of a partitioned matrix and the inverse parallel transport $\boldsymbol{R}^{\top}_{\boldsymbol{\xi}(\boldsymbol{s})\to\boldsymbol{\mu}^O}$ to move the resulting covariance to $\mathcal{T}_{\boldsymbol{\xi}(\boldsymbol{s})}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\xi}(\boldsymbol{s})}\mathcal{M}$.

B. Movement Primitive Prediction through Conditioning

Observing the similarities between GPs with noisy observations [6] and KMPs, we reformulate KMPs to model a noisy function $\boldsymbol{\xi}(\boldsymbol{s}^*) = f(\boldsymbol{s}^*) + \boldsymbol{\delta}$ with process mean function value $\boldsymbol{e}$ and $\boldsymbol{\delta} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{s}^*})$. The prior on these noisy observations then becomes $\mathrm{cov}(\boldsymbol{\xi}(\boldsymbol{s}^*)) = \boldsymbol{K} + \lambda\boldsymbol{\Sigma}$ where $\boldsymbol{\Sigma} = \mathrm{blockdiag}(\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \ldots, \boldsymbol{\Sigma}_N)$ for the training data $\mathcal{D}$ (Section III). Note that similarly, parametric noise models have been used before [41]. This allows us to write

$$\begin{bmatrix}\boldsymbol{\mu} \\ \boldsymbol{\xi}(\boldsymbol{s}^*)\end{bmatrix} = \mathcal{N}\left(\boldsymbol{e}, \begin{bmatrix}\boldsymbol{K}+\lambda\boldsymbol{\Sigma} & \boldsymbol{k}^* \\ \boldsymbol{k}^{*\top} & \boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}^*)\end{bmatrix}\right) \tag{38}$$

which can be solved similarly as before through minimizing

$$\begin{aligned}J(\boldsymbol{\xi}(\boldsymbol{s}^*)) = c + \mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e})^{\top}\boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}) + \\ \frac{1}{2}\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e})^{\top}\boldsymbol{\Lambda}^{OO}\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e}).\end{aligned} \tag{39}$$

Note that, unlike for Gaussian conditioning, we assume both $\boldsymbol{\Sigma}$ and the kernel function result to be in $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$ and $\mathcal{T}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}\mathcal{M}$, respectively. This holds trivially for $\boldsymbol{\Sigma}$ as the Riemannian Gaussian Distribution which is used as the probabilistic reference expresses its covariance in $\mathcal{T}_{\boldsymbol{\mu}}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$. Due to this assumption, unlike the derivation in Section III-A where basis functions were assumed to be expressed in $\mathcal{T}_{\boldsymbol{e}}\mathcal{M} \otimes \mathcal{T}_{\boldsymbol{e}}\mathcal{M}$, we do not require a parallel transport in the following equations. Please see Section IV how this property is achieved for kernels. The simplified Taylor expansion (34) then evaluates to

$$\begin{aligned}J(\boldsymbol{\xi}(\boldsymbol{s}^*)) \approx c + \left(\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e}) - \boldsymbol{\epsilon}\right)^{\top}\boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}) + \\ \frac{1}{2}\left(\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e}) - \boldsymbol{\epsilon}\right)^{\top}\boldsymbol{\Lambda}^{OO}\left(\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e}) - \boldsymbol{\epsilon}\right)\end{aligned} \tag{40}$$

as the update $\boldsymbol{\epsilon}$ does not require parallel transport. Note the minus in front of $\boldsymbol{\epsilon}$, caused by $\boldsymbol{\epsilon}$ updating $\boldsymbol{\xi}(\boldsymbol{s}^*)$ which is the base of the Log. The update evaluates to

$$\begin{aligned}\boldsymbol{\epsilon} &= \left(\boldsymbol{\Lambda}^{OO}\right)^{-1}\left(\boldsymbol{\Lambda}^{OO}\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e}) + \boldsymbol{\Lambda}^{OI}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e})\right) \\ &= \mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\boldsymbol{e}) - \boldsymbol{k}^*\left(\boldsymbol{K}+\lambda\boldsymbol{\Sigma}\right)^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}),\end{aligned} \tag{41}$$

Algorithm 3 Removing measurements in a cKMP.

for i in *measurements to remove* **do**
  $\boldsymbol{s} \leftarrow [\boldsymbol{s} \setminus \boldsymbol{s}_i]$, $\boldsymbol{\mu} \leftarrow [\boldsymbol{\mu} \setminus \boldsymbol{\mu}_i]$, $\boldsymbol{\Sigma} \leftarrow [\boldsymbol{\Sigma} \setminus \boldsymbol{\Sigma}_i]$ ▷ remove
  $\boldsymbol{K} \leftarrow [\boldsymbol{K} \setminus \boldsymbol{K}_{i,j}]$ ▷ remove
end for
$\boldsymbol{L} \leftarrow \boldsymbol{P} \cdot \boldsymbol{L}$ ▷ reorder using $\boldsymbol{P}$
while $\boldsymbol{L}$ not lower-triangular **do**
  $i, j \leftarrow \max(j - i | \boldsymbol{L}_{ij} \neq 0)$ ▷ Furthest off-diagonal
  $\boldsymbol{L} \leftarrow \boldsymbol{L} \cdot \boldsymbol{G}(i, j, \theta)$ ▷ Zero out $\boldsymbol{L}_{ij}$ using (47) – (49)
end while

using the same relations as in (35) and (36) for Gaussian conditioning. Note that when starting from an initial estimate $\boldsymbol{\xi}(\boldsymbol{s}^*) = \boldsymbol{e}$, convergence is given after one step through

$$\boldsymbol{\xi}(\boldsymbol{s}^*) = \mathrm{Exp}_{\boldsymbol{e}}\left(-\boldsymbol{k}^*\left(\boldsymbol{K}+\lambda\boldsymbol{\Sigma}\right)^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e})\right). \tag{42}$$

The covariance evaluates to

$$\boldsymbol{\Sigma}(\boldsymbol{s}^*) = \left(\boldsymbol{\Lambda}^{OO}\right)^{-1} = \boldsymbol{k}(\boldsymbol{s}^*, \boldsymbol{s}^*) - \boldsymbol{k}^*\left(\boldsymbol{K}+\lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{k}^{*\top} \tag{43}$$

which is very similar to the result for Gaussian conditioning in (37) but without parallel transport.

APPENDIX B
SUPPLEMENTAL DERIVATIONS

A. Soft Null Space Adaptations in Riemannian KMPs

In this section, we derive the Riemannian null space prediction (15). Integrating the penalty term $\beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}})^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}})$ to keep weights $\boldsymbol{\mu}_w$ close to desired ones $\hat{\boldsymbol{w}}$ in (9), we obtain

$$\begin{aligned}J(\boldsymbol{\mu}_w) \approx \sum_{n=1}^{N}\Big(\left(\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w + \mathrm{Log}_{\boldsymbol{\mu}_n}\boldsymbol{e}\right)^{\top}\boldsymbol{\Sigma}_n^{-1} \\ \left(\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s}_n)^{\top}\boldsymbol{\mu}_w + \mathrm{Log}_{\boldsymbol{\mu}_n}\boldsymbol{e}\right)\Big) \\ + \alpha\boldsymbol{\mu}_w^{\top}\boldsymbol{\mu}_w + \beta(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}})^{\top}(\boldsymbol{\mu}_w - \hat{\boldsymbol{w}}).\end{aligned} \tag{44}$$

Setting the derivative of this cost function to zero, we obtain

$$\begin{aligned}\left(\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s})\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}_{\parallel}^{\top}(\boldsymbol{s}) + (\alpha+\beta)\,\boldsymbol{I}\right)\boldsymbol{\mu}_w^* = \\ -\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s})\boldsymbol{\Sigma}^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}) + \beta\hat{\boldsymbol{w}}\end{aligned} \tag{45}$$

Defining $\lambda = \alpha + \beta$, this is the same result as (10) except for the additional term $\beta\hat{\boldsymbol{w}}$. As (45) is similar to the Euclidean derivations of [10], we can thus follow the steps laid out therein and in Section III-A, arriving at the optimal solution

$$\begin{aligned}\boldsymbol{\mu}_w^* = &-\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s})\left(\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s})^{\top}\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s}) + \lambda\boldsymbol{\Sigma}\right)^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{e}) \\ &+ \frac{\beta}{\lambda}\left[\boldsymbol{I} - \boldsymbol{\Phi}_{\parallel}(\boldsymbol{s})\left(\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s})^{\top}\boldsymbol{\Phi}_{\parallel}(\boldsymbol{s}) + \lambda\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Phi}_{\parallel}^{\top}(\boldsymbol{s})\right]\hat{\boldsymbol{w}}.\end{aligned}$$

From the desired modification $\hat{\boldsymbol{\xi}} = \mathrm{Exp}_{\boldsymbol{e}}\left(\hat{\boldsymbol{\Phi}}_{\parallel}^{\top}\hat{\boldsymbol{w}}\right)$ resulting in weights $\hat{\boldsymbol{w}} = \hat{\boldsymbol{\Phi}}_{\parallel}\left(\hat{\boldsymbol{\Phi}}_{\parallel}^{\top}\hat{\boldsymbol{\Phi}}_{\parallel}\right)^{-1}\Gamma_{\boldsymbol{e}\to\boldsymbol{\xi}(\boldsymbol{s}^*)}\left(\mathrm{Log}_{\boldsymbol{\xi}(\boldsymbol{s}^*)}(\hat{\boldsymbol{\xi}})\right)$, we kernelize the solution to obtain (15), assuming $\frac{\beta}{\lambda} = 1$ without loss of generality – this factor can, e.g., be pulled into $\hat{\boldsymbol{w}}$.

B. Restoring the Lower-triangular Form of $\boldsymbol{L}$

When removing a measurement (Section V-C) in the Cholesky-factorized KMP (cKMP) before the end, the decomposition $\boldsymbol{L}$ needs reordering. While still a valid decomposition,

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Transactions on Robot Learning. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TRL.2026.3756270

the permuted[6] $\boldsymbol{L}' = \boldsymbol{P}\boldsymbol{L}$ is not lower-triangular anymore in case we want to remove one or more measurements before the last measurement. Instead, it has rows "sticking out", e.g.,

$$\boldsymbol{L}' = \begin{pmatrix} l_{1,1} & \dots & 0 & \dots & 0 \\ & \ddots & & & \\ \dots & & 0 & \dots & 0 \\ l_{m,1} & \dots & l_{m,m} & l_{m,m+1} & 0 \\ & & & \ddots & \\ \dots & \dots & \dots & & 0 \\ l_{n,1} & \dots & \dots & l_{n,n-1} & 0 \end{pmatrix}, \tag{46}$$

where entries $l_{m,m+1}$ (or in general, $l_{m,m+k}$) extend beyond the lower triangle. After removing the last rows and columns, $\boldsymbol{L}'$ is unusable for forward- / backward-substitution as it would result in an overdetermined system of equations. Entries reaching over the diagonal can be zeroed out using Givens rotations ($c_\theta = \cos(\theta)$, $s_\theta = \sin(\theta)$) [42, Ch. 12.5]

$$\boldsymbol{G}(i,j,\theta) = \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & c_\theta & s_\theta & \dots & 0 \\ 0 & \dots & -s_\theta & c_\theta & \dots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 1 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}. \tag{47}$$

When applied on the right side, i.e. $\boldsymbol{L}^* = \boldsymbol{L} \cdot \boldsymbol{G}(i,j,\theta)$, they act on the row $\boldsymbol{L}_i = (*, \dots, *, \boldsymbol{L}_{i,j-1}, \boldsymbol{L}_{i,j}, *, \dots, *)$, leading to an equation for the element we would like to zero out

$$\boldsymbol{L}^*_{i,j} = -s_\theta \boldsymbol{L}_{i,j-1} + c_\theta \boldsymbol{L}_{i,j} \stackrel{!}{=} 0. \tag{48}$$

We can thus determine the entries of the Givens rotation to

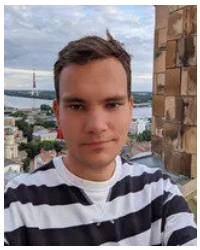
$$r = \sqrt{\boldsymbol{L}^2_{i,j-1} + \boldsymbol{L}^2_{i,j}}, \quad c_\theta = \frac{\boldsymbol{L}_{i,j-1}}{r}, \quad s_\theta = \frac{\boldsymbol{L}_{i,j}}{r}, \tag{49}$$

where we have used the trigonometric identity $c_\theta^2 + s_\theta^2 = 1$. These rotations have to be applied to all row vectors in rows $[i, \dots, n]$; we can omit applying it to the last rows we want to remove. Sorting rows to be processed in order $\max(k); \min(i)$, where k is the number of elements the row reaches over the diagonal, we can restore the lower triangular shape of $\boldsymbol{L}'$. The full algorithm is outlined in Algorithm 3. The runtime depends on which measurement we want to remove, as the removed and all following rows have to be partially recomputed – removing towards the end is thus progressively cheaper. As mentioned in Section V-C, for removing measurements at the end we can simply drop rows, making the Cholesky decomposition especially appealing for continuously updating via points as shown in Section VI-D.

[6]Note that the permutation $\boldsymbol{P}\boldsymbol{L}$ can be implemented by reordering the matrix, not requiring $\mathcal{O}(N^3)$ matrix multiplications.



Maximilian Mühlbauer received his B.Sc. and M.Sc. degrees in Mechanical Engineering respectively Robotics from the Technical University of Munich (TUM). He is a research scientist in the department of Cognitive Robotics at the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR) and a Ph.D. student at the Technical University of Munich. His research focuses on probabilistic machine learning and control methods for human-robot collaboration with applications in space, medical and industrial robotics.



Arne Sachtler received the M.Sc. degree in robotics, cognition, intelligence from the Technical University of Munich (TUM), Munich, Germany, in 2020. He is currently working toward the Ph.D. degree with the Institute of Robotics and Mechatronics, German Aerospace Center, Weßling, Germany, and the School of Computation, Information and Technology, Technical University of Munich. His research interests include application of differential geometry to robotics.



Markus Knauer is a final-year Ph.D. student at the Technical University of Munich (TUM), Munich, Germany, and a Researcher with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany. His research interests include data-efficient robot skill learning and adaptation, interactive imitation learning, and foundation models for robot manipulation.



Cem Küçükgenç received his B.Sc. degree in Mechanical Engineering from the Middle East Technical University (METU) in 2022, and his M.Sc. degree in Mechatronics, Robotics and Biomechanical Engineering from the Technical University of Munich (TUM) in 2025. During his M.Sc., he worked as a student researcher and completed his master's thesis in the Department of Cognitive Robotics at the Institute of Robotics and Mechatronics, German Aerospace Center (DLR). His research focused on robot skill learning and geometric methods for learning from demonstration. He is currently a Robotics Software Engineer, developing intelligent robotic systems for industrial applications.



Yanlong Huang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. He carried out his research as a Post-Doctoral Researcher with the Max Planck Institute for Intelligent Systems and Italian Institute of Technology. He is currently an Associate Professor with the School of Computer Science, University of Leeds, Leeds, U.K. His research interests include imitation learning and robot manipulation. He serves as an Associate Editor for IEEE Transactions on Robotics.



Alin Albu-Schäffer received his M.S. in electrical engineering from the Technical University of Timisoara, Romania in 1993 and his Ph.D. in automatic control from the Technical University of Munich in 2002. Since 2012 he is the head of the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR). Moreover, he is a professor at the Technical University of Munich, holding the Chair for "Sensor Based Robotic Systems and Intelligent Assistance Systems" at the School of Computation, Information and Technology. His personal research interests include robot design, modeling and control, nonlinear control, flexible joint and variable compliance robots, impedance and force control, physical human-robot interaction, bio-inspired robot design and control. He received several awards, including the IEEE King-Sun Fu Best Paper Award of the Transactions on Robotics in 2012 and 2014; several ICRA and IROS Best Paper Awards as well as the DLR Science Award. He was strongly involved in the development of the DLR light-weight robot and its commercialization through technology transfer to KUKA. He is the coordinator of euROBIN, the European network of excellence on intelligent robotics, IEEE Fellow and RAS-member.



João Silvério is a senior researcher and group leader at the German Aerospace Center (DLR) in Munich, where he leads the Interactive Skill Learning group since 2022. He received a PhD in Robotics from the Italian Institute of Technology (IIT), Genoa, in 2017. From 2017 to 2019, he was a postdoctoral researcher at IIT, and from 2019 to 2022, he held a postdoctoral position at the Idiap Research Institute in Martigny, Switzerland. His research focuses on machine learning for robotics, with particular interest in learning for control, imitation learning, and reinforcement learning.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/