Supplementary Material
Orientation Probabilistic Movement Primitives on Riemannian Manifolds

Leonel Rozo[1] **Vedant Dave**[1, 2]
[1]Bosch Center for Artificial Intelligence. Renningen, Germany.
[2]University of Leoben. Leoben, Austria.
leonel.rozo@de.bosch.com vedant.dave@unileoben.ac.at

Abstract: This document contains: (*i*) a notation table describing the most important variables and parameters used across the paper and this supplementary material, (*ii*) algorithms summarizing the learning process of both classic ProMPs and our Riemannian approach, and (*iii*) a detailed analysis of the experiments where we provide hyperparameters values and compare our approach against Euclidean (geometry-unaware) ProMP formulations. A video of our experiments can be watched at https://sites.google.com/view/orientation-promp.

1 Background

1.1 Notation

Symbols	Description
$\mathcal{M}$	Riemannian manifold
$\mathcal{T}_{\mathbf{p}}\mathcal{M}$	Tangent space of manifold $\mathcal{M}$ at $\mathbf{p} \in \mathcal{M}$
$\mathcal{T}\mathcal{M}$	Tangent bundle (group of all the tangent vectors in $\mathcal{M}$)
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$
$\mathcal{N}_{\mathcal{M}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Riemannian Gaussian distribution with mean $\boldsymbol{\mu} \in \mathcal{M}$ and covariance $\boldsymbol{\Sigma} \in \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$
$\boldsymbol{\theta}$	Parameters of Gaussian distribution
$\text{Exp}_{\mathbf{x}}(\cdot)$	Exponential map at $\mathbf{p} \in \mathcal{M}$
$\text{Log}_{\mathbf{x}}(\cdot)$	Logarithmic map at $\mathbf{p} \in \mathcal{M}$
$\Gamma_{\mathbf{x} \to \mathbf{y}}(\cdot)$	Parallel Transport from $\mathcal{T}_{\mathbf{p}}\mathcal{M}$ to $\mathcal{T}_{\mathbf{q}}\mathcal{M}$
$\mathcal{S}^m$	m-dimensional sphere manifold
$\phi_i(z_t)$	Normalized Gaussian basis function at time phase z_t
$\boldsymbol{w}$	ProMP weight vector
$\boldsymbol{\Psi}_t$	ProMP basis function matrix at z_t

Table 1: Notation for ProMP and Riemannian manifolds.

1.2 ProMPs algorithm

When learning from demonstrations, the example trajectories often differ in time length. ProMP overcomes this issue by introducing a phase variable z to decouple the data from the time instances, which in turn allows for temporal modulation. In this case, the demonstration ranges from $z_0 = 0$ to $z_T = 1$, redefining the demonstrated trajectory as $\boldsymbol{\tau} = \{\boldsymbol{y}_t\}_{t=z_0}^{z_T}$. The basis functions that form $\boldsymbol{\Psi}$ are defined as a function of the phase variable z. Specifically, ProMP uses Gaussian basis functions for stroke-based movements, defined as

$$b_m(z_t) \;=\; \exp\left(\frac{-(z_t - c_m)^2}{2h}\right) \quad \forall m = 1, \ldots, N_\phi, \tag{1}$$

5th Conference on Robot Learning (CoRL 2021), London, UK.

Algorithm 1: Classic ProMP Learning

Data: Set of N demonstrations with observations $\boldsymbol{Y}_n \quad \forall n = 1, \ldots, N$.
Input : Number of basis functions N_ϕ, width h, and regularization term λ.
Output: Mean $\boldsymbol{\mu_w}$ and covariance $\boldsymbol{\Sigma_w}$ of $\mathcal{P}(\boldsymbol{w}; \boldsymbol{\theta})$.
foreach *demonstration* n **do**
  Compute phase variables: $z_n = \frac{t_n}{t_n^{final}}$.
  Compute basis functions matrix $\boldsymbol{\Psi}_t$ using $\boldsymbol{\phi}_m(z_t)$ to build the basis matrix $\boldsymbol{\Psi}$.
  Compute weight vector $\boldsymbol{w}_n$ via $\boldsymbol{w}_n = (\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Psi} + \lambda \mathbf{I})^{-1}\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{Y}_n$.
Fit a Gaussian over the weight vectors $\boldsymbol{w}_n$:

$$\boldsymbol{\mu_w} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{w}_n, \; \boldsymbol{\Sigma_w} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{w}_n - \boldsymbol{\mu_w})(\boldsymbol{w}_n - \boldsymbol{\mu_w})^\mathsf{T}$$

return $\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}$

with width h and center c_i, which are often experimentally designed. These Gaussian basis functions are then normalized, leading to $\phi_m(z_t) = \frac{b_m(z_t)}{\sum_{j=1}^{N_\phi} b_j(z_t)}$. The classic ProMP learning process is summarized in Algorithm 1.

1.3 Riemannian manifolds

Table 2 provides the different expressions for the Riemannian distance, exponential and logarithmic maps, and parallel transport operation for the sphere manifold $\mathcal{S}^m$. Note that these expressions slightly differ from those used in [1], which actually correspond to exponential and logarithmic maps defined using Lie Theory [2]. The main difference lies on the fact that the Lie-based maps are defined with respect to the identity element of the Lie group (the manifold origin, loosely speaking). Therefore, they are coupled with transport operations from and to the origin in order to be applied on the complete manifold. Here we use maps that are defined at any point $\boldsymbol{x} \in \mathcal{M}$. Further details can be found when analyzing how the retraction operation is defined using Lie and Riemannian manifolds theories [2, 3].

Operation	**Formula**
$d_{\mathcal{M}}(\mathbf{x}, \mathbf{y})$	$\arccos(\mathbf{x}^\mathsf{T}\mathbf{y})$
$\text{Exp}_{\mathbf{x}}(\boldsymbol{u})$	$\mathbf{x}\cos(\|\boldsymbol{u}\|) + \overline{\boldsymbol{u}}\sin(\|\boldsymbol{u}\|)$ with $\overline{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$
$\text{Log}_{\mathbf{x}}(\mathbf{y})$	$d_{\mathcal{M}}(\mathbf{x}, \mathbf{y}) \frac{\mathbf{y} - \mathbf{x}^\mathsf{T}\mathbf{y}\,\mathbf{x}}{\|\mathbf{y} - \mathbf{x}^\mathsf{T}\mathbf{y}\,\mathbf{x}\|}$
$\Gamma_{\mathbf{x}\to\mathbf{y}}(\boldsymbol{v})$	$\Big(-\mathbf{x}\sin(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^\mathsf{T} + \overline{\boldsymbol{u}}\cos(\|\boldsymbol{u}\|)\overline{\boldsymbol{u}}^\mathsf{T} + (\boldsymbol{I} - \overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}^\mathsf{T})\Big)\boldsymbol{v}$ with $\overline{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}$ and $\boldsymbol{u} = \text{Log}_{\mathbf{x}}(\mathbf{y})$

Table 2: Principal operations on $\mathcal{S}^m$ (see Absil et al. [4] for details).

We will also need to parallel transport symmetric positive-definite matrices $\mathbf{V}$ (e.g. covariance matrices) as follows: $\Gamma_{\mathbf{x}\to\mathbf{y}}(\mathbf{V}) = \Gamma_{\mathbf{x}\to\mathbf{y}}(\boldsymbol{l_v})^\mathsf{T}\Gamma_{\mathbf{x}\to\mathbf{y}}(\boldsymbol{l_v})$, where $\boldsymbol{l_v} = \text{vec}(\boldsymbol{L_v})$ and $\mathbf{V} = \boldsymbol{L_v}^\mathsf{T}\boldsymbol{L_v}$.

2 Orientation ProMPs

2.1 Computation of the marginal distribution

The marginal distribution of $\mathbf{y}_t$ can be computed as[1]

$$\mathcal{P}(\mathbf{y}; \boldsymbol{\theta}) = \int \mathcal{N}_{\mathcal{M}}(\mathbf{y}|\underbrace{\text{Exp}_{\mathbf{p}}(\boldsymbol{\Psi w})}_{\boldsymbol{\mu_y}}, \boldsymbol{\Sigma_y})\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})d\boldsymbol{w}, \tag{2}$$

where the marginal distribution depends on two probability distributions that lie on different manifolds. However, the mean $\boldsymbol{\mu_y}$ depends on a single fixed point $\mathbf{p} \in \mathcal{M}$, and $\boldsymbol{\mu_w} \in \mathcal{T}_{\mathbf{p}}\mathcal{M}$. We exploit

[1] We drop time index t for the sake of notation.

these two observations to solve the marginal (2) on the tangent space $\mathcal{T}_{\mathbf{p}}\mathcal{M}$ as follows

$$\begin{aligned}\mathcal{P}(\text{Log}_{\mathbf{p}}(\mathbf{y})) &= \int \mathcal{N}(\text{Log}_{\mathbf{p}}(\mathbf{y})|\boldsymbol{\Psi}\boldsymbol{w}, \tilde{\boldsymbol{\Sigma}}_{\mathbf{y}})\mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}})d\boldsymbol{w}, \\ &= \mathcal{N}(\text{Log}_{\mathbf{p}}(\mathbf{y})|\boldsymbol{\Psi}\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Psi}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^{\mathsf{T}} + \tilde{\boldsymbol{\Sigma}}_{\mathbf{y}}),\end{aligned}$$

where $\tilde{\boldsymbol{\Sigma}}_{\mathbf{y}} = \Gamma_{\boldsymbol{\mu}_{\mathbf{y}}\to\mathbf{p}}(\boldsymbol{\Sigma}_{\mathbf{y}})$ is the parallel-transported covariance $\boldsymbol{\Sigma}_{\mathbf{y}}$ from $\boldsymbol{\mu}_{\mathbf{y}}$ to $\mathbf{p}$. Note that this marginal distribution still lies on the tangent space $\mathcal{T}_{\mathbf{p}}\mathcal{M}$, so we need to map it back to $\mathcal{M}$ using the exponential map, which leads to the final marginal

$$\mathcal{P}(\mathbf{y};\boldsymbol{\theta}) = \mathcal{N}_{\mathcal{M}}(\mathbf{y}|\underbrace{\text{Exp}_{\mathbf{p}}(\boldsymbol{\Psi}\boldsymbol{\mu}_{\boldsymbol{w}})}_{\hat{\boldsymbol{\mu}}_{\mathbf{y}}}, \hat{\boldsymbol{\Sigma}}_{\mathbf{y}}), \tag{3}$$

where $\hat{\boldsymbol{\Sigma}}_{\mathbf{y}} = \Gamma_{\mathbf{p}\to\hat{\boldsymbol{\mu}}_{\mathbf{y}}}(\boldsymbol{\Psi}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^{\mathsf{T}} + \tilde{\boldsymbol{\Sigma}}_{\mathbf{y}})$ is the parallel transportation of the full covariance matrix of the final marginal.

2.2 Gradient-based optimization for MGLM

The geodesic regression problem

$$(\hat{\mathbf{p}}, \hat{\boldsymbol{u}}) = \underset{(\mathbf{p},\boldsymbol{u})\in\mathcal{T}\mathcal{M}}{\text{argmin}} \frac{1}{2}\sum_{i=1}^{T} d_{\mathcal{M}}(\hat{\mathbf{y}}_i, \mathbf{y}_i)^2 \tag{4}$$

does not yield an analytical form like the original ProMP. As explained by Fletcher [5], a solution can be obtained through gradient descent, which requires to compute the derivative of the Riemannian distance function and the derivative of the exponential map. The latter is split into derivatives w.r.t the initial point $\mathbf{p}$ and the initial velocity $\boldsymbol{u}$. These gradients can be computed in terms of Jacobi fields (i.e., solutions to a second order equation subject to certain initial conditions under a Riemannian curvature tensor [5]).

However, to solve the multilinear geodesic regression

$$(\hat{\mathbf{p}}, \hat{\boldsymbol{u}}_j) = \underset{(\mathbf{p},\boldsymbol{u}_j)\in\mathcal{T}\mathcal{M}\,\forall j}{\text{argmin}} \frac{1}{2}\sum_{i=1}^{T} d_{\mathcal{M}}(\hat{\mathbf{y}}_i, \mathbf{y}_i)^2, \quad \text{with } \hat{\mathbf{y}}_i = \text{Exp}_{\mathbf{p}}(\boldsymbol{U}\boldsymbol{x}_i), \tag{5}$$

Kim et al. [6] compute the corresponding gradients by exploiting the insight that the adjoint operators resemble parallel transport operations. In such a way, we can overcome the hurdle of designing special adjoint operators for the multivariate case, and instead, perform parallel transport operations to approximate the necessary gradients. This multivariate framework serves the purpose of our first objective, namely, compute the weight vector for each demonstration lying on a Riemannian manifold $\mathcal{M}$. The weight estimate is here obtained by leveraging (5), leading to

$$(\hat{\mathbf{p}}, \hat{\boldsymbol{w}}_m) = \underset{(\mathbf{p},\boldsymbol{w}_m)\in\mathcal{T}\mathcal{M}\,\forall m}{\text{argmin}} \underbrace{\frac{1}{2}\sum_{t=1}^{T} d_{\mathcal{M}}(\text{Exp}_{\mathbf{p}}(\boldsymbol{W}\boldsymbol{\phi}_t), \mathbf{y}_t)^2}_{E(\mathbf{p},\boldsymbol{w}_m)}, \tag{6}$$

where $\boldsymbol{\phi}_t$ is the vector of basis functions at time t and $\boldsymbol{W}$ contains the set of estimated tangent weight vectors $\hat{\boldsymbol{w}}_m \in \mathcal{T}_{\hat{\mathbf{p}}}\mathcal{M}$ (i.e., N_ϕ tangent vectors emerging out from the point $\mathbf{p} \in \mathcal{M}$), that is, $\boldsymbol{W} = \text{blockdiag}(\hat{\boldsymbol{w}}_1^{\mathsf{T}}, \ldots, \hat{\boldsymbol{w}}_{N_\phi}^{\mathsf{T}})$.

To solve (6), we need to compute the gradients of $E(\mathbf{p}, \boldsymbol{w}_m)$ with respect to $\mathbf{p}$ and each $\boldsymbol{w}_m$. As stated above, these gradients depend on the so-called adjoint operators, which broadly speaking, bring each error term $\text{Log}_{\hat{\mathbf{y}}_t}(\mathbf{y}_t)$ from $\mathcal{T}_{\hat{\mathbf{y}}_t}\mathcal{M}$ to $\mathcal{T}_{\mathbf{p}}\mathcal{M}$, with $\hat{\mathbf{y}}_t = \text{Exp}_{\mathbf{p}}(\boldsymbol{W}\boldsymbol{\phi}_t)$. Therefore, these adjoint operators can be approximated as parallel transport operations as proposed in [6]. This leads to the following reformulation of the error function of (6)

$$E(\mathbf{p}, \boldsymbol{w}_m) = \frac{1}{2}\sum_{t=1}^{T} \|\Gamma_{\hat{\mathbf{y}}_t\to\mathbf{p}}(\text{Log}_{\hat{\mathbf{y}}_t}(\mathbf{y}_t))\|^2. \tag{7}$$

Algorithm 2: Orientation ProMP Learning

Data: Set of N demonstrations with quaternion data $\boldsymbol{Y}_n = \{\mathbf{y}_t\}_{t=1}^{T} \quad \forall n = 1, \ldots, N$, with $\mathbf{y}_t \in \mathcal{S}^3$.
Input : Number of basis functions N_ϕ, width h, learning rate η, maximum learning rate $\eta_{\max}$.
Output: Tangent space origin $\mathbf{p}$, mean $\boldsymbol{\mu}_w$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{w}}$ of $\mathcal{P}(\boldsymbol{w}; \boldsymbol{\theta})$.
foreach *demonstration* n **do**
1. Compute phase variables: $z_n = \frac{t_n}{t_n^{final}}$.
2. Compute basis functions matrix $\boldsymbol{\Psi}_t$ using $\boldsymbol{\phi}_m(z_t)$ to build the basis matrix $\boldsymbol{\Psi}$ of (3).
3. **Gradient descent for MGLM**:
Initialize: $\mathbf{p}_0 \in \mathcal{M}$ (only for $n = 1$) and weight vectors $\boldsymbol{w}_{m,0}^{(n)} \quad \forall m = 1, \ldots, N_\phi$.
Compute estimated trajectory points $\hat{\mathbf{y}}_t = \mathrm{Exp}_{\mathbf{p}}(\boldsymbol{W}\boldsymbol{\phi}_t)$.
while *termination criteria* **do**
$\hat{\mathbf{p}} = \mathrm{Exp}_{\mathbf{p}_k}(-\eta \nabla_{\mathbf{p}} E)$ using (8).
$\hat{\boldsymbol{w}}_m = \Gamma_{\mathbf{p}_k \to \mathbf{p}_{k+1}}(\boldsymbol{w}_{m,k} - \eta \nabla_{\boldsymbol{w}_m} E)$ using (8).
if $E(\hat{\mathbf{p}}, \hat{\boldsymbol{w}}_m) < E(\mathbf{p}_k, \boldsymbol{w}_{m,k})$ **then**
$\mathbf{p}_{k+1} \leftarrow \hat{\mathbf{p}}$
$\boldsymbol{w}_{m,k+1} \leftarrow \hat{\boldsymbol{w}}_m$
$\eta = \min(2\eta, \eta_{\max})$
else
$\eta = \eta/2$

Fit a Gaussian over the set of concatenated weight vectors $\{\boldsymbol{w}^{(n)}\}_{n=1}^{N}$ with $\boldsymbol{w}^{(n)} = \left[\boldsymbol{w}_1^{(n)^\mathsf{T}} \ldots \boldsymbol{w}_{N_\phi}^{(n)^\mathsf{T}}\right]^\mathsf{T}$:

$$\boldsymbol{\mu}_{\boldsymbol{w}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{w}^{(n)}, \; \boldsymbol{\Sigma}_{\boldsymbol{w}} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{w}^{(n)} - \boldsymbol{\mu}_w)(\boldsymbol{w}^{(n)} - \boldsymbol{\mu}_w)^\mathsf{T}$$

return $\mathbf{p}, \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}$

Then, the approximated gradients of the error function $E(\mathbf{p}, \boldsymbol{w}_m)$ correspond to

$$\nabla_{\mathbf{p}} E(\mathbf{p}, \boldsymbol{w}_m) \approx -\sum_{t=1}^{T} \Gamma_{\hat{\mathbf{y}}_t \to \mathbf{p}}(\mathrm{Log}_{\hat{\mathbf{y}}_t}(\mathbf{y}_t)), \text{ and } \nabla_{\boldsymbol{w}_m} E(\mathbf{p}, \boldsymbol{w}_m) \approx -\sum_{t=1}^{T} \boldsymbol{\phi}_{t,m} \Gamma_{\hat{\mathbf{y}}_t \to \mathbf{p}}(\mathrm{Log}_{\hat{\mathbf{y}}_t}(\mathbf{y}_t)). \tag{8}$$

The learning process for orientation ProMPs on quaternion trajectories is given in Algorithm 2.

2.3 Blending

Classic ProMP blends a set of movement primitives by using a product of Gaussian distributions. When it comes to blend primitives in $\mathcal{M}$, one needs to consider that each trajectory distribution is parametrized by a set of weight vectors that lie on different tangent spaces $\mathcal{T}_{\mathbf{p}}\mathcal{M}$. We resort to the Gaussian product formulation on Riemannian manifolds introduced by Zeestraten [7], where the log-likelihood of the product is iteratively maximized using a gradient-based approach as proposed by Dubbelman [8]. Formally, the log-likelihood of a product of Riemannian Gaussian distributions is given by (factoring out constant terms)

$$\ell(\mathbf{y}) = -\frac{1}{2}\sum_{s=1}^{S} \mathrm{Log}_{\boldsymbol{\mu}_{\mathbf{y},s}}(\mathbf{y})^\mathsf{T} \boldsymbol{\Sigma}_{\mathbf{y},s}^{-1} \mathrm{Log}_{\boldsymbol{\mu}_{\mathbf{y},s}}(\mathbf{y}), \tag{9}$$

where $\boldsymbol{\mu}_{\mathbf{y},s}$ and $\boldsymbol{\Sigma}_{\mathbf{y},s}$ are the parameters of the marginal distribution $\mathcal{P}_s(\mathbf{y}; \boldsymbol{\theta})$ for the skill s. Note that the logarithmic maps in (9) act on different tangent spaces $\mathcal{T}_{\boldsymbol{\mu}_{\mathbf{y},s}}\mathcal{M}, \forall\, s = 1 \ldots S$. In order to perform the log-likelihood maximization, we need to switch the base and argument of the maps while ensuring that the original log-likelihood function remains unchanged. To do so, we can leverage the relationship $\mathrm{Log}_{\mathbf{x}}(\mathbf{y}) = -\mathrm{Log}_{\mathbf{y}}(\mathbf{x})$ as well as parallel transport operations to overcome this problem, leading to

$$\mathcal{J} = \frac{1}{2}\sum_{s=1}^{S} \mathrm{Log}_{\boldsymbol{\mu}^+}(\boldsymbol{\mu}_{\mathbf{y},s})^\mathsf{T} \boldsymbol{\Lambda}_{\mathbf{y},s} \mathrm{Log}_{\boldsymbol{\mu}^+}(\boldsymbol{\mu}_{\mathbf{y},s}) \tag{10}$$

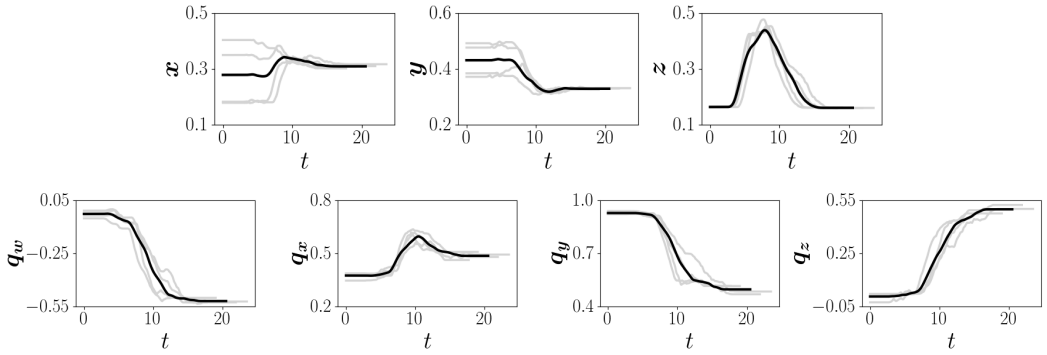

Figure 1: Time-series plot of the `re-orient` skill demonstrations (—), and mean trajectory (—) of the marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta})$. *Top*: end-effector position variables $[x, y, z]$ given in m. *Bottom*: end-effector orientation represented as a quaternion $[\boldsymbol{q}_w, \boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z]$. Time axis given in sec.

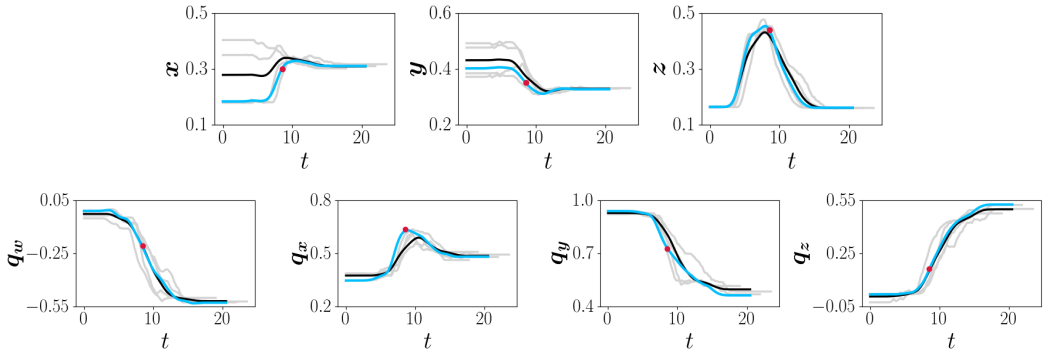

Figure 2: Time-series plot of the `re-orient` skill demonstrations (—), original mean trajectory (—) of the marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta})$, and resulting mean trajectory (—) of the new marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta}^*)$ passing through a given via-point (•) $\boldsymbol{p}^* \in \mathbb{R}^3 \times \mathcal{S}^3$. *Top*: end-effector position variables $[x, y, z]$ given in m. *Bottom*: end-effector orientation represented as a quaternion $[\boldsymbol{q}_w, \boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z]$. Time axis given in sec.

where $\boldsymbol{\mu}^+$ is mean of the resulting Gaussian (that we are estimating) and $\boldsymbol{\Lambda}_{\mathbf{y},s} = \Gamma_{\boldsymbol{\mu}_{\mathbf{y},s}\to\boldsymbol{\mu}^+}(\boldsymbol{\Sigma}^{-1}_{\mathbf{y},s})$. We can rewrite (10) by defining the vector $\epsilon(\boldsymbol{\mu}^+) = \left[\mathrm{Log}_{\boldsymbol{\mu}^+}(\boldsymbol{\mu}_{\mathbf{y},1})^\mathsf{T} \cdots \mathrm{Log}_{\boldsymbol{\mu}^+}(\boldsymbol{\mu}_{\mathbf{y},S})^\mathsf{T}\right]^\mathsf{T}$ and the block diagonal matrix $\boldsymbol{\Lambda} = \mathrm{blockdiag}(\boldsymbol{\Lambda}_{\mathbf{y},1}, \cdots, \boldsymbol{\Lambda}_{\mathbf{y},S})$. This results in $\mathcal{J}$ having the form of the objective function used to compute the empirical mean $\boldsymbol{v}$ of a Gaussian distribution on a Riemannian manifold $\mathcal{M}$ (Eq. 2.115 in Dubbelman [8]),

$$\mathcal{J}(\boldsymbol{v}) = \frac{1}{2}\epsilon(\boldsymbol{v})^\mathsf{T}\boldsymbol{\Lambda}\epsilon(\boldsymbol{v}), \tag{11}$$

from which is possible to iteratively compute the mean as

$$\boldsymbol{v}_{k+1} \leftarrow \mathrm{Exp}_{\boldsymbol{v}_k}(\Delta_{\boldsymbol{v}}) \text{ with } \Delta_{\boldsymbol{v}} = -(\boldsymbol{J}^\mathsf{T}\boldsymbol{\Lambda}\boldsymbol{J})^{-1}\boldsymbol{J}^\mathsf{T}\boldsymbol{\Lambda}\epsilon(\boldsymbol{v}), \tag{12}$$

where $\boldsymbol{J}$ is the Jacobian of $\epsilon(\boldsymbol{v})$ with respect to the basis of the tangent space of $\mathcal{M}$ at $\boldsymbol{v}_k$.

We can now carry out a similar iterative estimation of the mean $\boldsymbol{\mu}^+$ as follows

$$\Delta_{\boldsymbol{\mu}_k^+} = \left(\sum_{s=1}^{S}\alpha_s\boldsymbol{\Lambda}_{\mathbf{y},s}\right)^{-1}\left(\sum_{s=1}^{S}\alpha_s\boldsymbol{\Lambda}_{\mathbf{y},s}\mathrm{Log}_{\boldsymbol{\mu}_k^+}(\boldsymbol{\mu}_{\mathbf{y},s})\right), \quad \text{and} \quad \boldsymbol{\mu}^+_{k+1} \leftarrow \mathrm{Exp}_{\boldsymbol{\mu}_k^+}(\Delta_{\boldsymbol{\mu}_k^+}), \tag{13}$$

where $\boldsymbol{\Lambda}_{\mathbf{y},s} = \Gamma_{\boldsymbol{\mu}_{\mathbf{y},s}\to\boldsymbol{\mu}_k^+}(\boldsymbol{\Sigma}^{-1}_{\mathbf{y},s})$. After convergence at iteration K, we obtain the final parameters of the distribution $\mathcal{P}(\boldsymbol{y}^+) = \mathcal{N}_{\mathcal{M}}(\boldsymbol{y}^+|\boldsymbol{\mu}^+, \boldsymbol{\Sigma}^+)$ as follows

$$\boldsymbol{\mu}^+ \leftarrow \boldsymbol{\mu}^+_K \quad \text{and} \quad \boldsymbol{\Sigma}^+ = \left(\sum_{s=1}^{S}\alpha_s\boldsymbol{\Lambda}_{\mathbf{y},s}\right)^{-1}. \tag{14}$$

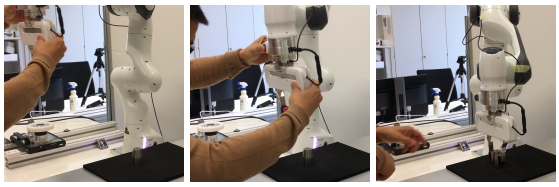

Figure 3: Snapshots of the human demonstrations of the `top-grasp` skill [9].

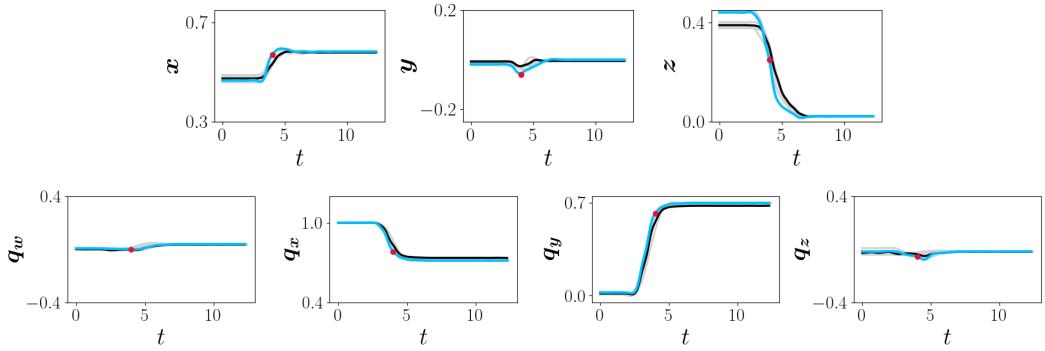

Figure 4: Time-series plot of the `top-grasp` skill demonstrations (—), original mean trajectory (—) of the marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta})$, and resulting mean trajectory (—) of the new marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta}^*)$ passing through a given via-point (•) $\boldsymbol{p}^* \in \mathbb{R}^3 \times \mathcal{S}^3$. *Top*: end-effector position variables $[x, y, z]$ given in m. *Bottom*: end-effector orientation represented as a quaternion $[\boldsymbol{q}_w, \boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z]$. Time axis given in sec.

3 Experiments

In this section, we provide additional details about the experiments carried out on both the synthetic dataset and the real manipulation skills reported in the main paper. Specifically, we provide the hyperparameters values used to train the models, as well as additional results regarding the comparison against different classical ProMPs implementations: Euler-angles and unit-norm approximations.

3.1 Synthetic data experiment on $\mathcal{S}^2$

The original trajectories were generated in $\mathbb{R}^2$ and subsequently projected to $\mathcal{S}^2$ by a simple mapping to unit-norm vectors. Each letter in the dataset was demonstrated $N = 8$ times. We trained 4 ProMP models, one for each letter of the set $\{\mathsf{G}, \mathsf{I}, \mathsf{J}, \mathsf{S}\}$. The models trained for I and J used $N_\phi = 30$ basis functions with width $h = 0.01$ and uniformly-distributed centers c, while the models trained over the datasets of more involved letters, i.e. G and S, employed $N_\phi = 60$ basis functions with width $h = 0.001$. All ProMP models were trained following Algorithm 2 with initial learning rate $\eta = 0.005$, and corresponding upper bound $\eta_{\max} = 0.03$.

3.2 Manipulation skills on $\mathbb{R}^3 \times \mathcal{S}^3$

Riemannian ProMPs. We collected a set of 4 demonstrations of the `re-orient` skill [9] through kinesthetic teaching, where full-pose robot end-effector trajectories $\{\boldsymbol{p}_t\}_{t=1}^T$ were recorded. Here $\boldsymbol{p}_t \in \mathbb{R}^3 \times \mathcal{S}^3$ represents the end-effector pose at time step t. The raw data was used to train a Riemannian ProMP on $\mathbb{R}^3 \times \mathcal{S}^3$, where the position and orientation models were learned using Algorithms 1 and 2, respectively. Both models used the same set of basis functions with $N_\phi = 40$, width $h = 0.02$, and uniformly-distributed centers c. The orientation model was trained using $\eta = 0.005$ as initial learning rate, and corresponding upper bound $\eta_{\max} = 0.03$.

Figure 1 shows the recorded demonstrations and the mean of the resulting marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta})$. It is evident that the ProMP model properly captures the relevant motion pattern on $\mathbb{R}^3 \times \mathcal{S}^3$. We then evaluated how this learned skill may adapt to a specific via-point. To do so, we chose a via point $\boldsymbol{p}^* \in \mathbb{R}^3 \times \mathcal{S}^3$, representing a new position and orientation of the end-effector at $t = 8.5\,\text{sec}$. By using the approach described in Section 3 of the main paper, we computed a new marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta}^*)$, where the updated mean is required to pass through $\boldsymbol{p}^*$. Figure 2 displays the

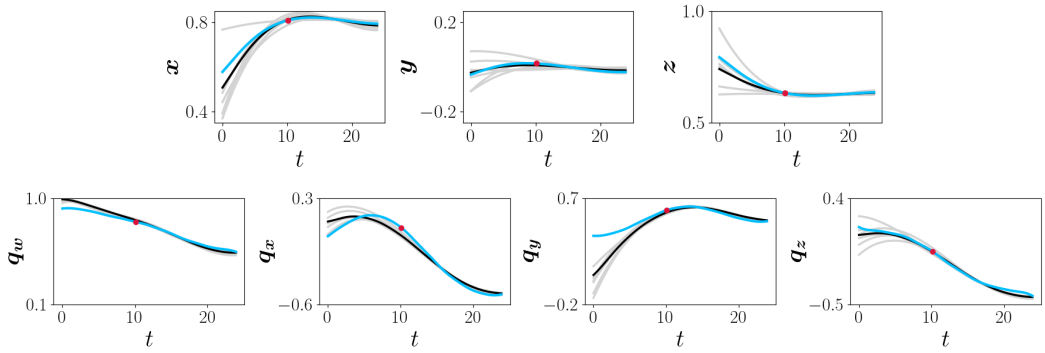

Figure 5: Time-series plot of the `key-turning` skill demonstrations (—), original mean trajectory (—) of the marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta})$, and resulting mean trajectory (—) of the new marginal distribution $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta}^*)$ passing through a given via-point (•) $\boldsymbol{p}^* \in \mathbb{R}^3 \times \mathcal{S}^3$. *Top*: end-effector position variables $[x, y, z]$ given in m. *Bottom*: end-effector orientation represented as a quaternion $[\boldsymbol{q}_w, \boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z]$. Time axis given in sec.

	Riemannian ProMPs	**Euler ProMPs**	**Unit-norm ProMPs**
Jerkiness	**21.1**	338.8	278.7
Tracking accuracy	$\mathbf{6.25 \times 10^{-5}}$	1.28×10^{-3}	6.24×10^{-4}
Deviation from mean	**18.16**	28.15	22.94

Table 3: Trajectory jerkiness (a.k.a smoothness [11]), tracking accuracy, and deviation from the mean trajectory for via-point adaptation of the `re-orient` skill. Bold values represent the best achieved result.

updated mean of the new marginal distribution, where the original trajectory mean is also displayed for reference. It can be observed that the updated trajectory successfully adapts to pass through the given via-point. Note that the adapted trajectory exploits the variability of the demonstration data (i.e. the associated covariance) to adapt the trajectory smoothly. This adaptation is more pronounced for the position trajectory (top row in Fig. 2), specially along the x axis.

To showcase the versatility of our approach, we trained two additional Riemannian ProMPs over data collected for the `top-grasp` and `key-turning` skills. The former corresponds to a dataset recorded on the same robotic setup as that of the `re-orient` skill [9], while the latter is a synthetic dataset recorded from a robotic simulator. Figure 3 displays the demonstration process of the `top-grasp` skill: Approach the metallic cap and grasp it from its top. Four demonstrations were used to train a Riemannian ProMP using a set of basis functions with $N_\phi = 20$, width $h = 0.025$, and uniformly-distributed centers c. Figure 4 shows our model successfully learns the relevant motion pattern of this skill. We also tested our approach on a dataset of synthetic demonstrations simulating a `key-turning` skill: Approach the key hole while aligning the end-effector accordingly, and later turning the end-effector 90 degrees clock-wise. Figure 5 displays the recorded demonstrations, the mean of the resulting marginal distribution, and updated mean trajectory for a given via-point. Our model once again captures the motion patterns for the `key-turning` skill, and exploits the demonstrations variability to update the mean trajectory while adapting to a via-point. These results confirm that our Riemannian formulation for ProMP makes full-pose trajectory learning and adaptation possible. The reproduction of the learned skills in simulation, using PyRoboLearn [10], can be watched in the supplementary video at https://sites.google.com/view/orientation-promp.

Comparison against Euler-angles and unit-norm ProMPs. Using the same demonstrations of the aforementioned skills, we trained three different ProMPs over the orientation trajectories to assess the importance of considering the quaternion geometry as proposed in this paper. The first model corresponds to our Riemannian ProMP approach, the second model is a classical ProMP trained over the Euler angles of the recorded orientation trajectories, and the third model is a classical ProMP trained over quaternion trajectories, whose output is normalized to comply with the unit-norm constraint. All models used the same set of hyperparameters detailed above. Moreover, our Riemannian models were trained using $\eta = 0.005$ as initial learning rate, and corresponding upper bound $\eta_{\max} = 0.03$. The output distributions from the Euler-angles model were transformed to

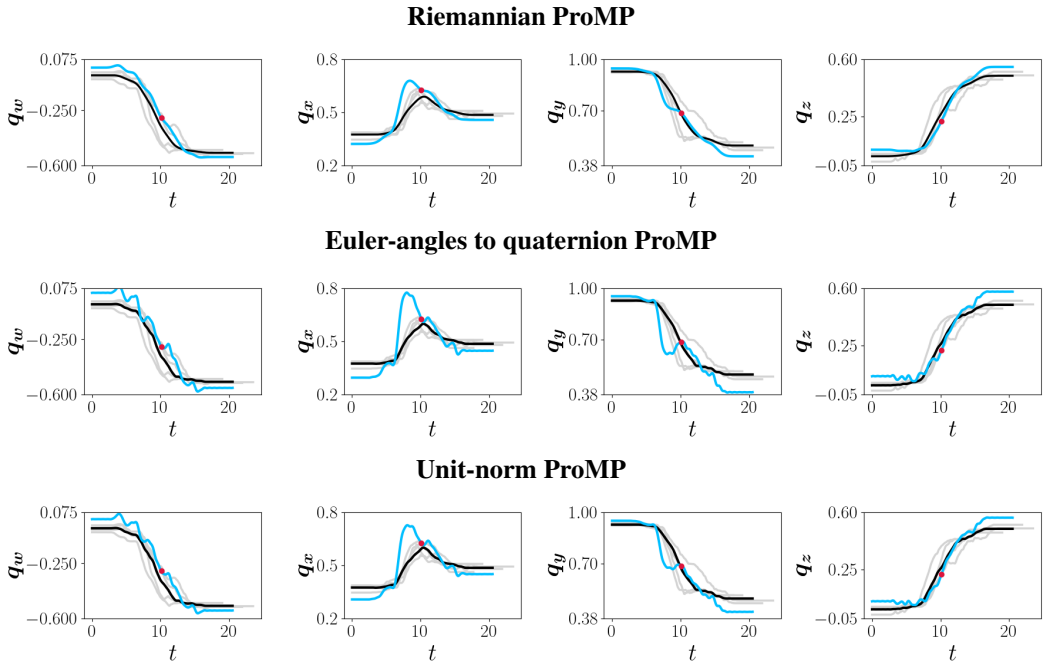

Figure 6: Comparison of mean trajectory retrieval and via-point adaptation for our approach (*top*), Euler-angles ProMPs (*middle*), and unit-norm approximation ProMPs (*bottom*). The figure shows time-series plots of the `re-orient` skill demonstrations (—), original mean trajectory (—) of the marginal $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta})$, and resulting mean trajectory (—) of the new marginal $\mathcal{P}(\boldsymbol{p};\boldsymbol{\theta}^*)$ passing through the via-point (•) $\boldsymbol{p}^* \in \times\mathcal{S}^3$. The end-effector orientation is represented as a quaternion $[\boldsymbol{q}_w, \boldsymbol{q}_x, \boldsymbol{q}_y, \boldsymbol{q}_z]$. Time axis given in sec.

	Riemannian ProMPs	**Euler ProMPs**	**Unit-norm ProMPs**
Jerkiness	$\mathbf{114.85}$	956.2	523.1
Tracking accuracy	$\mathbf{1.0 \times 10^{-7}}$	9.8×10^{-4}	2.3×10^{-4}
Deviation from mean	$\mathbf{2.43}$	5.2	4.1

Table 4: Trajectory jerkiness (a.k.a smoothness [11]), tracking accuracy, and deviation from the mean trajectory for via-point adaptation of the `top-grasp` skill. Bold values represent the best achieved result.

quaternion trajectories mainly for comparison purposes. However, the robot orientation controller also works with quaternion data, therefore such transformation is required whenever we need to send the desired orientation trajectory to the robot.

Figure 6 shows the mean of the resulting marginal (black line) and via-point adapted (light-blue line) distributions for each approach trained over the `re-orient` skill. No significant differences were observed when comparing the mean trajectories of the marginal distributions $\mathcal{P}(\mathbf{y};\boldsymbol{\theta})$ and $\mathcal{P}(\boldsymbol{y};\boldsymbol{\theta})$, i.e. Riemannian and classical ProMPs, respectively. However, when we evaluated how these models adapt to via-points (e.g. at $t = 10.0\,\text{sec}$), the importance of considering the quaternion space geometry is very noticeable. First, the deviation from the original mean $\sum_{t=1}^{T} d_{\mathcal{M}}(\mathbf{y}_t, \mathbf{y}_t^*)$ significantly increases for models trained over Euler angles and unit-norm approximation. Second, the tracking accuracy w.r.t the via-point $d_{\mathcal{M}}(\mathbf{y}_{t=10}, \mathbf{y}_{t=10}^*)$ is compromised when using the Euclidean approaches. Third, our Riemannian approach retrieves the smoothest adapted trajectories when compared to its Euclidean counterparts, which is of paramount important when controlling real robots (supplemental simulation videos using PyRoboLearn [10] can be found at https://sites.google.com/view/orientation-promp). Quantitative measures regarding trajectory smoothness, accuracy and deviation are given in Table 3, where it is clear that our Riemannian formulation outperforms the other two Euclidean methods. Similar results were obtained for both the `top-grasp` and `key-turning` skills, reported in Tables 4 and 5, respectively.

	Riemannian ProMPs	**Euler ProMPs**	**Unit-norm ProMPs**
Jerkiness	$\mathbf{2.52}$	28.8	18.9
Tracking accuracy	$\mathbf{7.0 \times 10^{-4}}$	2.8×10^{-3}	9.8×10^{-4}
Deviation from mean	$\mathbf{19.63}$	30.5	24.4

Table 5: Trajectory jerkiness (a.k.a smoothness [11]), tracking accuracy, and deviation from the mean trajectory for via-point adaptation of the `key-turning` skill. Bold values represent the best achieved result.

	Riemannian ProMPs	**Euler ProMPs**	**Unit-norm ProMPs**
Weights estimation	254.8 sec	1.8097 sec	3.2797 sec
Trajectory retrieval	0.2266 sec	0.2089 sec	0.2391 sec
Via-point conditioning	0.3261 sec	0.2321 sec	0.2481 sec

Table 6: Comparison of the computational cost when learning the model weights and retrieving the trajectory distribution for the standard and via-point cases of the `re-orient` skill. Trajectory retrieval and via-point conditioning times consider the computation cost over the whole trajectory.

As discussed in the main paper, the aforementioned benefits of our Riemannian approach comes at the cost of increasing the computational complexity of the model weights estimation. Table 6 reports the computational cost for our Riemannian formulation, the Euler and Unit-norm ProMPs for the `re-orient` skill. We measured the execution time for the model weights estimation, trajectory retrieval and via-point conditioning. It is evident that weights estimation based on multivariate geodesic regression takes significantly longer than the original ProMP approach. Note that this execution time can be easily reduced by relaxing the stopping criteria of the gradient-based optimizer, such as maximum iterations and geodesic reconstruction error. More importantly, the trajectory retrieval and via-point conditioning processes are not compromised, which are the methods that a practitioner may be interest in running on the real platform. All the approaches were implemented in unoptimized Python code.

References

[1] A. Ude, B. Nemec, T. Petrić, and J. Morimoto. Orientation in Cartesian space dynamic movement primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2997–3004, 2014. URL https://doi.org/10.1109/ICRA.2014.6907291.

[2] J. Solà, J. Deray, and D. Atchuthan. A micro lie theory for state estimation in robotics, 2020. URL https://arxiv.org/abs/1812.01537.

[3] N. Boumal. An introduction to optimization on smooth manifolds. Available online, Nov 2020. URL http://www.nicolasboumal.net/book.

[4] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2007. URL https://press.princeton.edu/absil.

[5] P. T. Fletcher. Geodesic regression and the theory of least squares on Riemannian manifolds. *International Journal of Computer Vision (IJCV)*, 105:171–185, 2013. URL https://doi.org/10.1007/s11263-012-0591-y.

[6] H. J. Kim, N. Adluru, M. D. Collins, M. K. Chung, B. B. Bendin, 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2705–2712, 2014. URL https://doi.org/10.1109/CVPR.2014.352.

[7] M. Zeestraten. *Programming by Demonstration on Riemannian Manifolds*. PhD thesis, University of Genova, Italy, 2018. URL https://iris.unige.it/handle/11567/930621#.YCgbRuoo85k.

[8] G. Dubbelman. *Intrinsic Statistical Techniques for Robust Pose Estimation*. PhD thesis, University of Amsterdam, Netherlands, 2011. URL https://hdl.handle.net/11245/1.348545.

[9] L. Rozo, M. Guo, A. G. Kupcsik, M. Todescato, P. Schillinger, M. Giftthaler, M. Ochs, M. Spies, N. Waniek, P. Kesper, and M. Bürger. Learning and sequencing of object-centric manipulation skills for industrial tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9072–9079, 2020. doi:10.1109/IROS45743.2020.9341570.

[10] B. Delhaisse, L. Rozo, and D. G. Caldwell. Pyrobolearn: A python framework for robot learning practitioners. In *Conference on Robot Learning (CoRL)*, pages 1348–1358, Osaka, Japan, October 2019. URL https://proceedings.mlr.press/v100/delhaisse20a.html.

[11] S. Balasubramanian, A. Melendez-Calderon, A. Roby-Brami, and E. Burdet. On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12(112):1–11, 2015. URL https://doi.org/10.1186/s12984-015-0090-9.