Learning Riemannian Manifolds for Geodesic Motion Skills

Hadi Beik-Mohammadi[1,2], Søren Hauberg[3], Georgios Arvanitidis[4], Gerhard Neumann[2], and Leonel Rozo[1]
[1]Bosch Center for Artificial Intelligence (BCAI), Renningen, Germany.
[2] Autonomous Learning Robots Lab, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany.
[3] Section for Cognitive Systems, Technical University of Denmark (DTU), Lyngby, Denmark.
[4] Max Planck Institute for Intelligent Systems (MPI-IS), Tübingen, Germany.
Emails: hadi.beik-mohammadi@de.bosch.com, sohau@dtu.dk, gear@tuebingen.mpg.de
gerhard.neumann@kit.edu, leonel.rozo@de.bosch.com

***Abstract*—For robots to work alongside humans and perform in unstructured environments, they must learn new motion skills and adapt them to unseen situations on the fly. This demands learning models that capture relevant motion patterns, while offering enough flexibility to adapt the encoded skills to new requirements, such as dynamic obstacle avoidance. We introduce a Riemannian manifold perspective on this problem, and propose to learn a Riemannian manifold from human demonstrations on which geodesics are natural motion skills. We realize this with a variational autoencoder (VAE) over the space of position and orientations of the robot end-effector. Geodesic motion skills let a robot plan movements from and to arbitrary points on the data manifold. They also provide a straightforward method to avoid obstacles by redefining the ambient metric in an online fashion. Moreover, geodesics naturally exploit the manifold resulting from multiple–mode tasks to design motions that were not explicitly demonstrated previously. We test our learning framework using a 7-DoF robotic manipulator, where the robot satisfactorily learns and reproduces realistic skills featuring elaborated motion patterns, avoids previously–unseen obstacles, and generates novel movements in multiple-mode settings.**

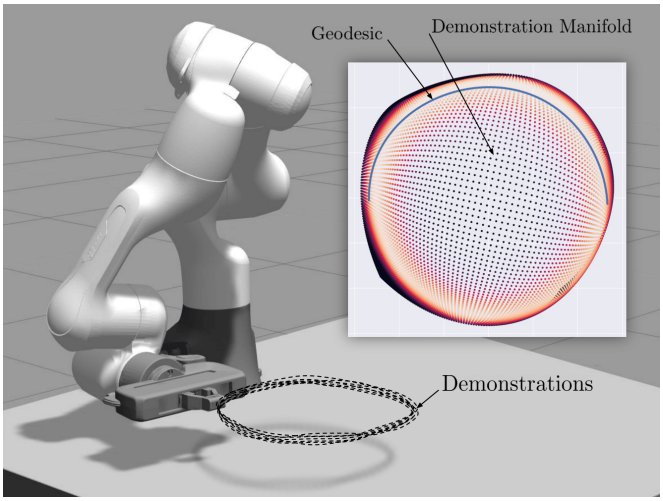

Fig. 1: From demonstrations we learn a variational autoencoder that spans a random Riemannian manifold. Geodesics on this manifold are viewed as motion skills.

I. Introduction

Robot motion generation has been actively investigated during the last decades, where motion planners and movement primitives have led to significant advances. When a robot moves in obstacle-free environments, the motion generation problem can be easily solved by off-the-shelf motion planners [13]. However, the problem is significantly more involved in unstructured environments when (static and dynamic) obstacles occupy the robot workspace [27]. Moreover, if the robot motion depends on variable targets, or requires to consider multiple-solution tasks, the motion generation problem exacerbates. Some of the aforementioned problems have been recently addressed from a learning-from-demonstration (LfD) perspective, where a skill model is learned by extracting the relevant motion patterns from human examples [28].

LfD approaches are advantageous as they do not necessarily require a model of the environment, and can easily adapt to variable targets on the fly [28]. Three main lines of work stand out in the LfD field, namely, *(1)* dynamical-system-based approaches [20] which focus on capturing motion dynamics, *(2)* probabilistic methods [6, 29, 19] which exploit data variability and model uncertainty, and more recently, *(3)* neural networks [37, 4] which address generalization problems. Despite their significant contributions (see Section II), several challenges are still open: encoding and reproduction of full-pose end-effector movements, skill adaptation to unseen or dynamic obstacles, handling multiple-solution (a.k.a. multiple-mode) tasks, generalization to unseen situations, among others.

In this paper we provide an LfD approach that addresses several of these problems, through a Riemannian perspective for learning robot motions from demonstrations. Unlike previous works [17, 26], where skill manifolds are built from locally smooth manifold learning [11], we leverage a Riemannian formulation. Specifically, we develop a variational autoencoder (VAE) that learns a Riemannian submanifold of $\mathbb{R}^3 \times \mathcal{S}^3$ from human demonstrations. We exploit geodesics (i.e. shortest paths) on this learned manifold as the robot motion generation mechanism, from which full-pose end-effector trajectories are generated.

These geodesics reproduce motions starting from arbitrary initial points on the manifold, and they can adapt to avoid dynamic obstacles in the robot environment by redefining the ambient metric.

Our approach can learn manifolds encoding multiple-solution tasks, from which novel (unobserved) robot motions may naturally arise when computing geodesics.

We illustrate our approach with a simple example, and we test our method in robotic experiments using a 7-DoF robotic manipulator.

The experiments show how our approach learns and reproduces realistic robot skills featuring complex motion patterns and obstacle avoidance. Moreover, we demonstrate how our approach can discover new solutions for unseen task setups in a multiple–mode setting.

In summary, we contribute a new view on motion skill learning that navigates along geodesics on a manifold learned via a novel VAE over position-orientation space. We show how this allows the robot to generate useful movements beyond the demonstration set, while avoiding dynamic obstacles that were not part of the learning procedure.

II. Background and related work

We first briefly review some relevant work on learning from demonstrations, variational autoencoders (VAEs), and Riemannian geometry. We also introduce some recent connections between VAEs and Riemannian geometry, which is the backbone of our work.

A. Learning robot motion from demonstrations

Learning from demonstrations (LfD) provides a framework for a robot to quickly learn tasks by *observing* several demonstrations provided by a (human) expert [28]. The demonstrations are then used to learn a model of the task (e.g., a movement primitive, a reward, or plan) which is then used to synthesize new robot motions [32]. In particular, movement primitives (MPs) describe complex motion skills, and represent an alternative solution to classic motion planners [13] for generating robot motions. We here exploit LfD to learn a Riemannian skill manifold, which we later employ to drive the robot motion generation.

LfD approaches can be categorized into: *(1)* dynamical-system-based approaches [20] which capture the demonstrated motion dynamics [36], *(2)* probabilistic methods [6, 29, 19] that take advantage of data variability and model uncertainty, and *(3)* neural networks [37, 4] aimed at generalization problems. Our method leverages a neural network (VAE) to learn a Riemannian metric that incorporates the network uncertainty. This metric allows us to generate motions that resemble the demonstrations. Unlike the aforementioned approaches, our method allows for online obstacle avoidance by rescaling the learned metric. Although obstacle avoidance might still be possible by defining via-points in methods like [29, 37, 19], this problem was not explicitly considered in any of them.

Finally, human demonstrations may show different solution trajectories for the same task [33, 37], which is often tackled through hierarchical approaches [24, 14]. In this context, our method permits to encode multiple-solution tasks into the learned Riemannian manifold, which is exploited to not only reproduce the demonstrated solutions but also to come up with a hybrid solution built on a synergy of a subset of the demonstrations. These novel solutions naturally arise from our geodesic motion generator. Note that previous learning frameworks generate robot motions that are restricted to the provided solutions for the task at hand.

B. Variational autoencoders (VAEs)

A variational autoencoder (VAE) [23] is a generative model that captures the data density $p(\boldsymbol{x})$ through a latent variable $\boldsymbol{z}$ that generally has a significantly lower dimension than $\boldsymbol{x}$. In the interest of simplicity, we consider Gaussian VAEs corresponding to the generative model

$$p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|\boldsymbol{0}, \mathbb{I}_d\right), \qquad \boldsymbol{z} \in \mathcal{Z}; \quad (1)$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|\mu_{\boldsymbol{\theta}}(\boldsymbol{z}), \mathbb{I}_D \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{z})\right), \qquad \boldsymbol{x} \in \mathcal{X}. \quad (2)$$

Here $\mu_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}$ and $\sigma_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathbb{R}^D_+$ are deep neural networks with parameters $\boldsymbol{\theta}$ that estimate the mean and the variance of the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$.

When the generative process is nonlinear, exact inference becomes intractable, and VAEs apply a variational approximation of the evidence (marginal likelihood). The corresponding evidence lower bound (ELBO) is then

$$\begin{aligned}\mathcal{L}_{ELBO} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}))\right] \\ - \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right),\end{aligned} \quad (3)$$

where $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\mu_{\boldsymbol{\phi}}(\boldsymbol{x}), \mathbb{I}_d \sigma^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$ approximates the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$ by two deep neural networks $\mu_{\boldsymbol{\phi}}(\boldsymbol{x}) : \mathcal{X} \to \mathcal{Z}$ and $\sigma_{\boldsymbol{\phi}}(\boldsymbol{x})) : \mathcal{X} \to \mathbb{R}^d_+$. This approximate posterior is often denoted the *inference* or *encoder* distribution, while the generative process $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is known as the *decoder*. As mentioned previously, we use VAEs to learn a robot skill.

C. Riemannian geometry

Riemannian manifolds can be intuitively seen as curved d-dimensional surfaces that are described by smoothly-varying positive-definite inner products, characterized by the Riemannian metric $\boldsymbol{M}$ [25]. These manifolds locally resemble a Euclidean space $\mathbb{R}^d$, and have a globally defined differential structure. For our purposes, it suffices to consider manifolds as defined by a mapping function

$$f : \mathcal{Z} \to \mathcal{X}, \quad (4)$$

where both $\mathcal{Z}$ and $\mathcal{X}$ are open subsets of Euclidean spaces with $\dim \mathcal{Z} < \dim \mathcal{X}$. We then say that $\mathcal{M} = f(\mathcal{Z})$ is a manifold immersed in the ambient space $\mathcal{X}$.

Given a curve $c : [0, 1] \to \mathcal{Z}$, we can measure its length on the manifold as

$$\mathcal{L}_c = \int_0^1 \|\partial_t f(c(t))\| \mathrm{d}t. \quad (5)$$

By applying the chain-rule, we see that this can be equivalently expressed as

$$\mathcal{L}_c = \int_0^1 \sqrt{\dot{c}(t)^\mathsf{T} \boldsymbol{M}(c(t)) \dot{c}(t)} \mathrm{d}t. \quad (6)$$

Here $\dot{c}_t = \partial_t c_t$ is the curve derivative and where we have introduced the *Riemannian metric*

$$\boldsymbol{M}(\boldsymbol{z}) = \boldsymbol{J}_f(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_f(\boldsymbol{z}), \tag{7}$$

where $\boldsymbol{J}_f$ is the Jacobian of f that we evaluate at $\boldsymbol{z} \in \mathcal{Z}$. We may think of the metric as forming a local inner product in $\mathcal{Z}$ that inform us how to measure lengths locally. This construction relies on the Euclidean metric of $\mathcal{X}$; we will later extend this to also form a Riemannian metric. Having defined a notion of *curve length* (5), we can trivially define shortest paths, or *geodesics*, as curves of minimal length. Geodesics are the generalization of straight lines on the Euclidean space to Riemannian manifolds. They will serve as our motion generation mechanism as explained in Section IV. Note that geodesics have been recently used as solutions of trajectory optimizers for quadrotors control [35].

D. Variational autoencoders as Riemannian manifolds

To make the link between VAEs and Riemannian geometry [1], we may write the generative process of a VAE (2) as

$$f_{\boldsymbol{\theta}}(\boldsymbol{z}) = \mu_{\boldsymbol{\theta}}(\boldsymbol{z}) + \text{diag}(\epsilon)\sigma_{\boldsymbol{\theta}}(\boldsymbol{z}), \quad \epsilon \sim \mathcal{N}(\boldsymbol{0}, \mathbb{I}_D). \tag{8}$$

This is also known as the *reparametrization trick* [23] and is illustrated in Fig. 2. We see that this is similar to the mapping function (4) that defined a manifold in the previous section. The difference being that now $f_{\boldsymbol{\theta}}$ is stochastic. We can rewrite this stochastic mapping as [12]

$$f_{\boldsymbol{\theta}}(\boldsymbol{z}) = \begin{pmatrix} \mathbb{I}_D, & \text{diag}(\epsilon) \end{pmatrix} \begin{pmatrix} \mu_{\boldsymbol{\theta}}(\boldsymbol{z}) \\ \sigma_{\boldsymbol{\theta}}(\boldsymbol{z}) \end{pmatrix} = \boldsymbol{P}\, g(\boldsymbol{z}), \tag{9}$$

where $\boldsymbol{P}$ is a random matrix, and $g(\boldsymbol{z})$ is the concatenation of $\mu_{\boldsymbol{\theta}}(\boldsymbol{z})$ and $\sigma_{\boldsymbol{\theta}}(\boldsymbol{z})$. In this notation, we can view the VAE as a random projection of a deterministic manifold spanned by g, and the metric under this mapping is

$$\bar{\boldsymbol{M}}(\boldsymbol{z}) = \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z}) + \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z}). \tag{10}$$

Geodesics under this metric have been shown to be faithful to the data used for training the VAE [1]. Hauberg [15] argues that this is due to the contribution from σ to the metric and that disregarding this term gives an almost flat manifold geometry.

As mentioned, the definition of curve length relies on the Euclidean metric of the ambient space $\mathcal{X}$, but this is not a strict requirement. Arvanitidis et al. [3] argue that there is value in giving the ambient space a manually defined *Riemannian* metric and including that into the definition of curve length. Curve length is then defined as

$$\mathcal{L}_c = \int_0^1 \sqrt{\dot{c}(t)^\mathsf{T} \boldsymbol{J}_{f_{\boldsymbol{\theta}}}(c(t))^\mathsf{T} \boldsymbol{M}_{\mathcal{X}}(f_{\boldsymbol{\theta}}(c(t))) \boldsymbol{J}_{f_{\boldsymbol{\theta}}}(c(t)) \dot{c}(t)} \mathrm{d}t, \tag{11}$$

where $\boldsymbol{M}_{\mathcal{X}}$ is the ambient space metric, which can now vary smoothly across $\mathcal{X}$. The corresponding Riemannian metric of $\mathcal{Z}$ is then

$$\begin{aligned} \bar{\boldsymbol{M}}(\boldsymbol{z}) = \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{M}_{\mathcal{X}}(\mu_{\boldsymbol{\theta}}(\boldsymbol{z})) \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z}) \\ + \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{M}_{\mathcal{X}}(\mu_{\boldsymbol{\theta}}(\boldsymbol{z})) \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z}). \end{aligned} \tag{12}$$

With this construction, it is straightforward to push geodesics away from certain regions of $\mathcal{X}$ by increasing $\boldsymbol{M}_{\mathcal{X}}$ there.

Note that geodesics do generally not follow a closed-form expression in these models, and numerical approximations are in order. This can be done by direct minimization of curve length [38, 22], A^* search [9], integration of the associated ODE [2], or various heuristics [8].

III. Riemannian manifold learning on $\mathbb{R}^3 \times \mathcal{S}^3$

Learning complex robot motion skills requires models that have enough capacity to learn and synthesize the relevant patterns of a motion while being flexible enough to adapt to new conditions. In this section, we describe how we tackle this problem by bringing a Riemannian manifold perspective to the robot learning problem. First, we explain how we exploit VAEs to access a low-dimensional learned manifold of the motion data where an ambient-space Riemannian metric is learned. This metric will be later used to generate robot motion trajectories via geodesics, as detailed in Section IV. In order to learn elaborated motion skills, which may display complex position and orientation trajectories, we represent the robot state as the full pose of the robot end-effector, i.e. its position $\boldsymbol{x} \in \mathbb{R}^3$ and orientation $\boldsymbol{q} \in \mathcal{S}^3$. We then seek a VAE that models a joint density of this state. We retain the usual Gaussian prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \mathbb{I}_d)$, but alter the generative distribution $p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{q}|\boldsymbol{z})$ to suit our needs. We will assume that position and orientation are conditionally independent,

$$p_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{q}|\boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z}), \tag{13}$$

such that all correlations between the two must be captured by the latent variable $\boldsymbol{z}$.

A. Position encoding on $\mathbb{R}^3$

To model the conditional distribution of end-effector positions $\boldsymbol{x}$, we opt for simplicity and choose this to be Gaussian,

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mu_{\boldsymbol{\theta}}(\boldsymbol{z}), \mathbb{I}_3 \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{z})), \tag{14}$$

where $\mu_{\boldsymbol{\theta}}$ and $\sigma_{\boldsymbol{\theta}}$ are neural networks parametrized by $\boldsymbol{\theta}$. One could argue that $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ should have zero probability mass outside the workspace of the robot, but we disregard such concerns as $\sigma^2_{\boldsymbol{\theta}}$ tends to take small values due to limited data noise. This implies that only a negligible fraction of the probability mass falls outside the robot workspace.

B. Orientation encoding on $\mathcal{S}^3$

Complex robot motions often involve elaborated orientation trajectories which require a suitable representation for motion learning and control. There exist several orientation representation such rotation matrices, Euler angles, and unit quaternions. Euler angles and rotation matrices are commonly used for reasons of simplicity. Unfortunately, Euler angles suffer from gimbal lock [18] which makes them an inadequate representation of orientation in robotics, and rotation matrices are a redundant representation requiring a high number of parameters. Unit quaternions are a convenient way to represent an orientation since they are compact, not redundant,

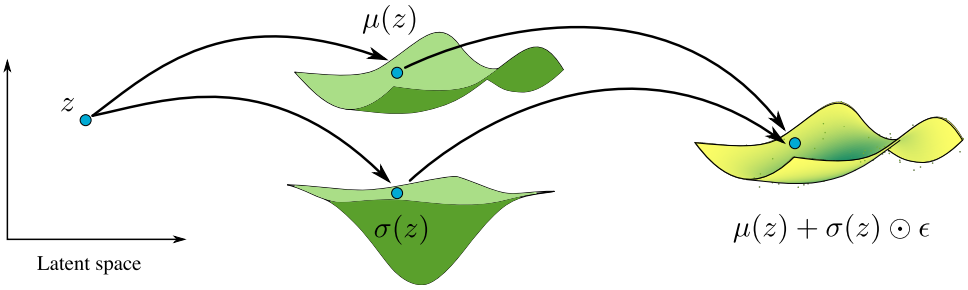

Fig. 2: In a Gaussian VAE, samples are generated by a random projection of the manifold jointly spanned by μ and σ.

and prevent gimbal lock. Also, they provide strong stability guarantees in closed–loop orientation control [7], they have been recently exploited in robot skills learning [34], and for data-efficient robot control tuning [21] under a Riemannian-manifold formulation.

We choose to represent orientations $\boldsymbol{q}$ as a unit quaternion, such that $\boldsymbol{q} \in \mathcal{S}^3$ with the additional antipodal identification that $\boldsymbol{q}$ and $-\boldsymbol{q}$ correspond to the same orientation. Formally, a unit quaternion $\boldsymbol{q}$ lying on the surface of a 3-sphere $\mathcal{S}^3$ can be represented using a 4-dimensional unit vector $\boldsymbol{q} = [q_w, q_x, q_y, q_z] \in \mathcal{S}^3$, where the scalar q_w and vector (q_x, q_y, q_z) represent the real and imaginary parts of the quaternion, respectively. To cope with antipodality, one could opt to model $\boldsymbol{q}$ as a point in a projective space, but for reasons of simplicity we let $\boldsymbol{q}$ live on the unit sphere $\mathcal{S}^3$. We then choose a generative distribution $p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z})$ such that $p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z}) = p_{\boldsymbol{\psi}}(-\boldsymbol{q}|\boldsymbol{z})$.

To construct a suitable distribution $p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z})$ over the unit sphere, we turn to the von Mises-Fischer (vMF) distribution, which is merely an isotropic Gaussian constrained to lie on the unit sphere [41]. This distribution is described by a mean direction $\boldsymbol{\mu}$ with $\|\boldsymbol{\mu}\| = 1$, and a concentration parameter $\kappa \geq 0$. Its density function takes the form

$$\mathrm{vMF}(\boldsymbol{q}|\boldsymbol{\mu}, \kappa) = C_D(\kappa) \exp\left(\kappa \boldsymbol{\mu}^\mathsf{T} \boldsymbol{q}\right), \qquad \|\boldsymbol{\mu}\| = 1, \tag{15}$$

where C_D is the normalization constant

$$C_D(\kappa) = \frac{\kappa^{\frac{D}{2}-1}}{(2\pi)^{\frac{D}{2}} I_{\frac{D}{2}-1}(\kappa)}, \tag{16}$$

with $I_{\frac{D}{2}-1}(\kappa)$ being the modified Bessel function of the first kind. Like the Gaussian, from which the distribution was constructed, the von Mises-Fischer distribution is unimodal. To build a distribution that is antipodal symmetric, i.e. $p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z}) = p_{\boldsymbol{\psi}}(-\boldsymbol{q}|\boldsymbol{z})$, we simply form a mixture of antipodal von Mises-Fischer distributions [16],

$$\begin{aligned} p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z}) = &\frac{1}{2}\mathrm{vMF}(\boldsymbol{q}|\boldsymbol{\mu}_{\boldsymbol{\psi}}(\boldsymbol{z}), \kappa_{\boldsymbol{\psi}}(\boldsymbol{z})) \\ &+ \frac{1}{2}\mathrm{vMF}(\boldsymbol{q}|-\boldsymbol{\mu}_{\boldsymbol{\psi}}(\boldsymbol{z}), \kappa_{\boldsymbol{\psi}}(\boldsymbol{z})), \end{aligned} \tag{17}$$

where $\boldsymbol{\mu}$ and κ are parametrized as neural networks. This mixture model is conceptually similar to a Bingham distribution [41], but is easier to implement numerically.

C. Variational inference

Our VAE model can be trained by maximizing the conventional evidence lower bound (ELBO) (3), which now is

$$\mathcal{L}_{ELBO} = \alpha \mathcal{L}_{\boldsymbol{x}} + \beta \mathcal{L}_{\boldsymbol{q}} - \mathrm{KL}\left(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z})\right), \tag{18}$$

$$\mathcal{L}_{\boldsymbol{x}} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right], \tag{19}$$

$$\mathcal{L}_{\boldsymbol{q}} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\psi}}(\boldsymbol{q}|\boldsymbol{z})\right], \tag{20}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ and $\boldsymbol{q} \in \mathcal{S}^3$ represent the position and quaternion of the end-effector, respectively. To balance the log-likelihood of position and orientation components, $\alpha > 0$ and $\beta > 0$ are proportionally scaled. Note that due to the antipodal nature of quaternions, raw demonstration data may contain negative or positive values for the same orientation. So, we avoid any pre-processing step of the data by considering two von Mises-Fischer distributions that encode the same orientation at both sides of the hypersphere. Practically, we double the training data, by including $\boldsymbol{q}_n$ and $-\boldsymbol{q}_n$ for all observations $\boldsymbol{q}_n$.

D. Induced Riemannian metric

Our generative process is parametrized by a set of neural networks. Specifically, $\mu_{\boldsymbol{\theta}}$ and $\sigma_{\boldsymbol{\theta}}$ are position mean and variance neural networks parameterized by $\boldsymbol{\theta}$, while $\mu_{\boldsymbol{\psi}}$ and $\kappa_{\boldsymbol{\psi}}$ are neural networks parameterized by $\boldsymbol{\psi}$ that represent the mean and concentration of the quaternion distribution. Following Sec. II-D the Jacobians of these functions govern the induced Riemannian metric as

$$\boldsymbol{M}(\boldsymbol{z}) = \boldsymbol{M}^{\boldsymbol{x}}_{\mu}(\boldsymbol{z}) + \boldsymbol{M}^{\boldsymbol{x}}_{\sigma}(\boldsymbol{z}) + \boldsymbol{M}^{\boldsymbol{q}}_{\mu}(\boldsymbol{z}) + \boldsymbol{M}^{\boldsymbol{q}}_{\kappa}(\boldsymbol{z}) \tag{21}$$

with

$$\begin{aligned} \boldsymbol{M}^{\boldsymbol{x}}_{\mu}(\boldsymbol{z}) &= \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z}), \quad \boldsymbol{M}^{\boldsymbol{x}}_{\sigma}(\boldsymbol{z}) = \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z}), \\ \boldsymbol{M}^{\boldsymbol{q}}_{\mu}(\boldsymbol{z}) &= \boldsymbol{J}_{\mu_{\boldsymbol{\psi}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_{\mu_{\boldsymbol{\psi}}}(\boldsymbol{z}), \quad \boldsymbol{M}^{\boldsymbol{q}}_{\sigma}(\boldsymbol{z}) = \boldsymbol{J}_{\sigma_{\boldsymbol{\psi}}}(\boldsymbol{z})^\mathsf{T} \boldsymbol{J}_{\sigma_{\boldsymbol{\psi}}}(\boldsymbol{z}), \end{aligned}$$

where $\boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}$, $\boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}$, $\boldsymbol{J}_{\mu_{\boldsymbol{\psi}}}$, $\boldsymbol{J}_{\sigma_{\boldsymbol{\psi}}}$ are the Jacobian of functions representing the position mean and variance, as well as the quaternion mean and concentration, respectively.

In practice, we want this metric $\boldsymbol{M}(\boldsymbol{z})$ to take large values in regions with little or no data, so that geodesics avoid passing through them. Following Arvanitidis et al. [1] we achieve this by using radial basis function (RBF) networks as our variance representation, whose kernels reliably extrapolate over the whole space. Since one of the main differences between Gaussian and von Mises-Fischer distributions lies on the way they represent data dispersion, the RBF network should consider a

reciprocal behavior when estimating variance for positions. In summary, the data uncertainty is encoded by the RBF networks representing $\sigma_{\boldsymbol{\theta}}^{-1}(\boldsymbol{z})$ and $\kappa_{\boldsymbol{\psi}}(\boldsymbol{z})$, which affect the Riemannian metric through their corresponding Jacobians as in (21).

IV. Geodesic Motion Skills

As mentioned previously, geodesics follow the trend of the data, and they are here exploited to reconstruct motion skills that resemble human demonstrations. Moreover, we explain how new geodesic paths, that avoid obstacles on the fly, can be obtained by a metric scaling process. In particular, we exploit ambient space metrics defined as a function of the obstacles configuration to locally deform the original learned Riemannian metric. Last but not least, our approach can encode multiple-solution skills, from which new hybrid trajectories (not previously shown to the robot) can be synthesized. We elaborate on each of these features in the sequel.

A. Geodesic motion generation

Geodesic curves generally follow the trend of the training data, due to the role of uncertainty in the metric. Specifically, Eq. (10) tells us that geodesics are penalized for crossing through regions where the VAE predictive uncertainty grows. This implies that if a set of demonstrations follows a circular motion pattern, geodesics starting from arbitrary points on the learned manifold will also generate a circular motion. This behavior is due to the way that the metric $\boldsymbol{M}$ is defined, as $\boldsymbol{M}$ is characterized by low values where data uncertainty is low (and vice-versa). Since the geodesics minimize the energy of the curve between two points on $\mathcal{M}$, which is calculated as a function of $\boldsymbol{M}$, they tend to stay on the learned manifold and avoid outside regions. This property makes us suggest that geodesics form a natural motion generation mechanism. Note that when using an Euclidean metric (i.e., an identity matrix), geodesics correspond to straight lines. Such geodesics certainly neglect the data manifold geometry.

Formally, we compute geodesics on $\mathcal{M}$ by approximating them by cubic splines $c \approx \omega_\lambda(\boldsymbol{z}_c)$, with $\boldsymbol{z}_c = \{\boldsymbol{z}_{c_0}, \dots, \boldsymbol{z}_{c_N}\}$, where $\boldsymbol{z}_{c_n} \in \mathcal{Z}$ is a vector defining a control point of the spline over the latent space $\mathcal{Z}$. Given N control points, $N-1$ cubic polynomials ω_{λ_i} with coefficients $\lambda_{i,0}$, $\lambda_{i,1}$, $\lambda_{i,2}$, $\lambda_{i,3}$ have to be estimated to minimize its Riemannian length

$$\mathcal{L}_{\omega_\lambda(\hat{c})} = \int_0^1 \sqrt{\langle \dot{\omega}_\lambda(\boldsymbol{z}_c), \boldsymbol{M}(\omega_\lambda(\boldsymbol{z}_c))\dot{\omega}_\lambda(\boldsymbol{z}_c)\rangle}\mathrm{d}t. \qquad (22)$$

Then, the final geodesic c computed in $\mathcal{Z}$ is used to generate the robot motion through the mean decoder networks $\mu_{\boldsymbol{\theta}}$ and $\mu_{\boldsymbol{\psi}}$. The resulting trajectory can be executed on the robot arm to reproduce the required skill.

To illustrate, we consider a simple experiment where the demonstration data at each time point is confined to $\mathbb{R}^2 \times \mathcal{S}^2$, i.e. only two dimensional positions and orientations are considered. We create synthetic position data that follows a J-shape and orientation data that follows a C-shape projected on the sphere (see Fig. 3). We fit our VAE model to this data, and visualize the corresponding latent space in Fig. 4.

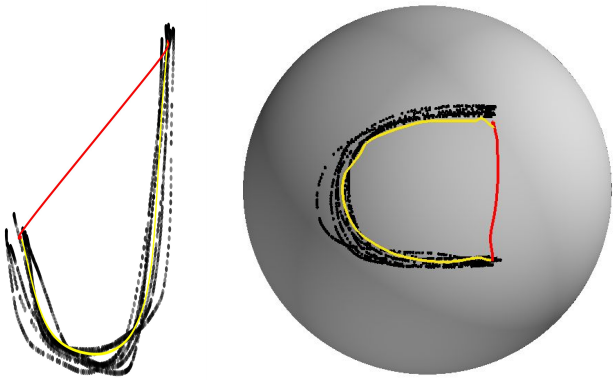

Fig. 3: As an illustration, we consider synthetic data that belong to $\mathbb{R}^2 \times \mathcal{S}^2$. The left panel depicts the J-shaped position data in $\mathbb{R}^2$ and the right panel shows the C-shaped orientation data on $\mathcal{S}^2$. The yellow and red curves show the geodesics computed based on Riemannian and Euclidean metrics, respectively.

Here the top panel shows the latent mean embeddings of the training data with a background color corresponding to the predictive uncertainty. We see low uncertainty near the data, and high otherwise. The bottom panel of Fig. 4 shows the same embedding but with a background color proportional to $\log\sqrt{\det \boldsymbol{M}}$. This quantity, known as the magnification factor [5], will generally take large values in regions where distances are large, implying that geodesics will try to avoid such regions. In the figure, we notice that the magnification factor is generally low, except on the 'boundary' of the data manifold, i.e. in regions where the predictive variance grows. Consequently, we observe that Riemannian geodesics (yellow curves in the figure) stay within the 'boundary' and are hence near the training data. In contrast, Euclidean geodesics (red curves in the figure) fail to stay in the data manifold. Our proposal is to use Riemannian geodesics to generate new motions for the robot.

B. Obstacle avoidance using ambient space metrics

Often human demonstrations do not include any notion of obstacles in the environment. As a result, obstacle avoidance is usually treated as an independent problem when generating robot motions in unstructured environments. A possible solution to integrate both problems is to provide obstacle–aware demonstrations, where the robot is explicitly taught how to avoid known obstacles. The main drawback here is that the robot is still unable to avoid unseen obstacles on the fly.

The Riemannian approach provides a natural solution to this problem. The natural metric in latent space (10) measures the length of a latent curve under the Euclidean space of the ambient space $\mathcal{X}$. We can easily modify this to take obstacles into account. Intuitively, we can increase the length of curves that intersect obstacles, such that geodesics are less likely to go near the obstacles. Formally, we propose to alter the ambient metric of the end-effector position to be

$$\boldsymbol{M}_{\mathcal{X}}^{\boldsymbol{x}}(\boldsymbol{x}) = \left(1 + \eta \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{o}\|^2}{2r^2}\right)\right)\mathbb{I}_3, \quad \boldsymbol{x} \in \mathbb{R}^3, \quad (23)$$

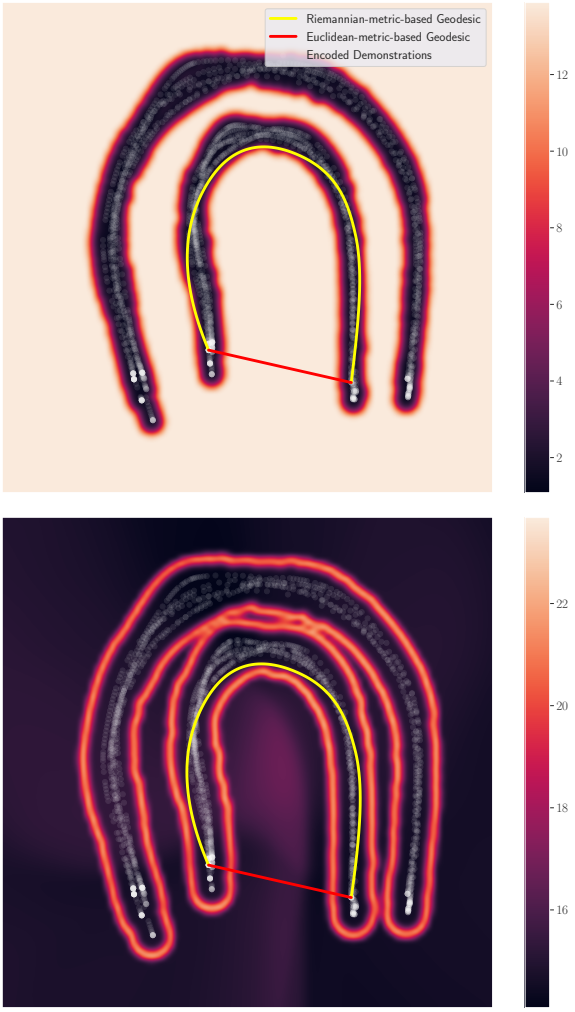

Fig. 4: *Top*: the variance measure, *bottom*: magnification factor of the Riemannian manifold learned from trajectories based on J and C English alphabet characters defined in $\mathbb{R}^2 \times \mathcal{S}^2$. The semi-transparent white points depict the encoded training set and the yellow curve depicts the geodesic in the latent space. The resulting manifold is composed of two similar clusters due to the antipodal encoding of the quaternions, where each cluster represents one side of the hyper-sphere. The yellow and red curves show the geodesics computed based on Riemannian and Euclidean metrics, respectively.

where $\eta > 0$ scales the cost, $\boldsymbol{o} \in \mathbb{R}^3$ and $r > 0$ represent the position and radius of the obstacle, respectively. For the orientation component, we use $\boldsymbol{M}^{\boldsymbol{q}}_{\mathcal{X}}(\boldsymbol{x}) = \mathbb{I}_4$. Under this ambient metric, geodesics will generally avoid the object, though we emphasize this is only a *soft* constraint. This approach is similar in spirit to CHOMP [31] except our formulation works along a low-dimensional learned manifold, whose solution is then projected to the task space of the robot. Under this ambient metric, the associated Riemannian metric of the latent space $\mathcal{Z}$ becomes

$$\boldsymbol{M}(\boldsymbol{z}) = \boldsymbol{M}^{\boldsymbol{x}}_{\mu}(\boldsymbol{z}) + \boldsymbol{M}^{\boldsymbol{x}}_{\sigma}(\boldsymbol{z}) + \boldsymbol{M}^{\boldsymbol{q}}_{\mu}(\boldsymbol{z}) + \boldsymbol{M}^{\boldsymbol{q}}_{\kappa}(\boldsymbol{z}), \qquad (24)$$

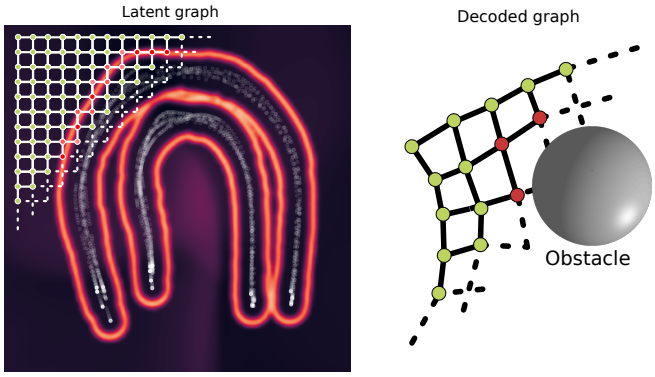

Fig. 5: Concept drawing. *Left:* The latent space is discretized to form a grid graph consisting of linearly spaced nodes, with edge weights matching Riemannian distances. *Right:* To efficiently handle obstacles, the graph is decoded, such that obstacles can easily be mapped to latent space.

$$\begin{aligned}
\text{with} \quad \boldsymbol{M}^{\boldsymbol{x}}_{\mu}(\boldsymbol{z}) &= \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}^{\boldsymbol{x}}_{\mathcal{X}}(\mu_{\boldsymbol{\theta}}(\boldsymbol{z})) \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z}), \\
\boldsymbol{M}^{\boldsymbol{x}}_{\sigma}(\boldsymbol{z}) &= \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}^{\boldsymbol{x}}_{\mathcal{X}}(\mu_{\boldsymbol{\theta}}(\boldsymbol{z})) \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z}), \\
\boldsymbol{M}^{\boldsymbol{q}}_{\mu}(\boldsymbol{z}) &= \boldsymbol{J}_{\mu_{\boldsymbol{\psi}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}^{\boldsymbol{q}}_{\mathcal{X}}(\mu_{\boldsymbol{\psi}}(\boldsymbol{z})) \boldsymbol{J}_{\mu_{\boldsymbol{\psi}}}(\boldsymbol{z}), \\
\boldsymbol{M}^{\boldsymbol{q}}_{\kappa}(\boldsymbol{z}) &= \boldsymbol{J}_{\kappa_{\boldsymbol{\psi}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}^{\boldsymbol{q}}_{\mathcal{X}}(\mu_{\boldsymbol{\psi}}(\boldsymbol{z})) \boldsymbol{J}_{\kappa_{\boldsymbol{\psi}}}(\boldsymbol{z}),
\end{aligned}$$

where $\boldsymbol{M}^{\boldsymbol{x}}_{\mathcal{X}}$ and $\boldsymbol{M}^{\boldsymbol{q}}_{\mathcal{X}}$ represent the position and orientation components of the obstacle–avoidance metric $\boldsymbol{M}_{\mathcal{X}}$, respectively. Here we emphasize that as the object changes position, the VAE does not need to be re-trained as the change is only in the ambient metric.

C. Real time motion generation

Having phrased motion generation as the computation of geodesics, we evidently need a fast and robust algorithm for computing geodesics. As we work with low–dimensional latent spaces, we here propose to simply discretize the latent space on a regular grid and use a graph–based algorithm for computing shortest paths.

Specifically, we create a uniform grid over the latent space, and assign a weight to each edge in the graph corresponding to the Riemannian distance between neighboring nodes (see Fig. 5). Geodesics are then found using Dijkstra's algorithm [10]. This algorithm selects a set of graph nodes,

$$G_c = \{\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{N-1}, \boldsymbol{g}_N\}, \quad \boldsymbol{g}_n \in \mathbb{R}^D,$$

where $\boldsymbol{g}_0$ and $\boldsymbol{g}_N$ represent the start and the target of the geodesic in the graph, respectively. To select these points, the shortest path on the graph is calculated by minimising the accumulated weight (cost) of each edge connecting two nodes calculated as in (6). To ensure a smooth trajectory, we fit a cubic spline ω_λ to the resulting set G_c by minimizing the mean square error. The spline computed in $\mathcal{Z}$ is finally used to generate the robot motion through the mean decoder networks $\mu_{\boldsymbol{\theta}}$ and $\mu_{\boldsymbol{\psi}}$. The resulting trajectory can be executed on the robot arm to reproduce the required skill.

One issue with this approach is that dynamic obstacles imply that geodesic distances between nodes may change dynamically. To avoid recomputing all edge weights every time an obstacle moves we do as follows. Since the learned

manifold does not change, we can keep a decoded version of the latent graph in memory (Fig. 5). This way we need not query the decoders at run-time. We can then find points on the decoded graph that are near obstacles and rescale their weights to take the ambient metric into account. Once the obstacle moves, we can reset the weights of points that are now sufficiently far from the obstacle. The center panel of Fig. 9 provides an example showing how the metric of the left panel is represented as a discrete graph.

V. Experiments

We evaluate the performance and capabilities of our method in two different scenarios[1]: a simulated pouring task, and a real-world grasping scenario both in $\mathbb{R}^3 \times \mathcal{S}^3$. In particular, the pouring task showcases a multiple-solution setting. For the experiments, we discuss the design of each experiment regarding manifold learning and geodesic computation. We also provide a visualization of the learned Riemannian metrics and geodesic representation in the latent space $\mathcal{Z}$. Furthermore, the code is available at: https://sites.google.com/view/geodesicmotion.

A. Setup description

We consider simulated and real robot demonstrations involving a 7-DoF Franka Emika Panda robot arm with a two–finger gripper. The demonstrations were recorded using kinesthetic teaching in the real grasping scenario meanwhile simulated pouring dataset was collected using the Franka ROS Interface [40] on Gazebo [39]. In both scenarios, the robot is controlled by an impedance controller.

We calculate geodesic on a 100×100 grid graph, and our straightforward Python implementation runs at 100Hz on ordinary PC hardware. The approach readily runs in real time.

B. VAE architecture

Our VAE architecture is implemented using PyTorch [30]. The decoder and encoder networks have two hidden layers with 200 and 100 neuron units. The same architecture is used for all the experiments. The RBF variance/concentration networks use 500 kernels calculated by k-means over the training dataset [1] and predefined bandwidth. The latent space of the VAE is 2-dimensional, while the ambient space is 7-dimensional, corresponding to $\mathbb{R}^3 \times \mathcal{S}^3$.

We employ a single neural network to represent both the position and orientation decoder means, meaning that our final metric is defined as

$$\boldsymbol{M}(\boldsymbol{z}) = \boldsymbol{M}_{\mu}^{\boldsymbol{x},\boldsymbol{q}}(\boldsymbol{z}) + \boldsymbol{M}_{\sigma}^{\boldsymbol{x}}(\boldsymbol{z}) + \boldsymbol{M}_{\kappa}^{\boldsymbol{q}}(\boldsymbol{z}), \tag{25}$$

$$\begin{aligned}
\text{with} \quad \boldsymbol{M}_{\mu}^{\boldsymbol{x},\boldsymbol{q}}(\boldsymbol{z}) &= \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}_{\mathcal{XQ}}(\boldsymbol{z}) \boldsymbol{J}_{\mu_{\boldsymbol{\theta}}}(\boldsymbol{z}), \\
\boldsymbol{M}_{\mathcal{XQ}}(\boldsymbol{z}) &= \begin{bmatrix} \boldsymbol{M}_{\mathcal{X}}(\mu_{\boldsymbol{\theta}}(\boldsymbol{z})) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{\mathcal{Q}}(\mu_{\boldsymbol{\psi}}(\boldsymbol{z})) \end{bmatrix}, \\
\boldsymbol{M}_{\sigma}^{\boldsymbol{x}}(\boldsymbol{z}) &= \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}_{\mathcal{X}}(\mu_{\boldsymbol{\theta}}^{\boldsymbol{x}}(\boldsymbol{z})) \boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}}(\boldsymbol{z}), \\
\boldsymbol{M}_{\kappa}^{\boldsymbol{q}}(\boldsymbol{z}) &= \boldsymbol{J}_{\kappa_{\boldsymbol{\psi}}}(\boldsymbol{z})^{\mathsf{T}} \boldsymbol{M}_{\mathcal{Q}}(\mu_{\boldsymbol{\psi}}(\boldsymbol{z})) \boldsymbol{J}_{\kappa_{\boldsymbol{\psi}}}(\boldsymbol{z}).
\end{aligned}$$

[1]We extensively evaluated our method on simulation settings as the COVID-19 pandemic prohibited access to our robotic labs. We still tested our learning approach with real robot data to validate that our insights apply in realistic scenarios. We plan to run more real experiments whenever possible.

where $\boldsymbol{J}_{\mu_{\boldsymbol{\theta}}} \in \mathbb{R}^{(D_{\mathcal{X}}+D_{\mathcal{Q}})\times d}$ is the Jacobian of the joint decoder mean network (position and quaternion), and $\boldsymbol{J}_{\sigma_{\boldsymbol{\theta}}} \in \mathbb{R}^{D_{\mathcal{X}}\times d}$ and $\boldsymbol{J}_{\sigma_{\boldsymbol{\psi}}} \in \mathbb{R}^{D_{\mathcal{Q}}\times d}$ are the Jacobians of the decoder variance and concentration RBF networks. Since the position and quaternion share the same decoder mean network, the output vector is split into two parts, accordingly. The quaternion part of the decoder mean is projected to the $\mathcal{S}^3$ to then define the corresponding von Mises-Fischer distribution (15).

The ELBO parameters α and β in (18) are found experimentally to guarantee good reconstruction of both position and quaternion data. It is worth pointing out that we manually provided antipodal quaternions during training, which leads to better latent space structures and reconstruction accuracy.

C. Real grasping task

The first set of experiments is based on a dataset collected while a human operator performs kinesthetic demonstrations of a grasping skill. This particular grasping motion includes a $90°$ rotation when approaching the object for performing a side grasp [34]. The demonstration trajectories start from different end-effector poses, and they reach the same target position with slightly different orientations.

To reproduce the grasping skill, we computed a geodesic in $\mathcal{Z}$ which leads to a continuous trajectory in $\mathcal{X}$, that closely reproduces the rotation pattern observed during demonstrations.

Figure 6 depicts the magnification factor related to the learned manifold. The semi–transparent white points correspond to the latent representation of the training set, and the yellow curves are geodesics between points assigned from the start and endpoints of the demonstrations. The left panel in Fig. 6 shows geodesics in two different scenarios: The ones in the top cluster start from different poses and end up at the same target, and the geodesics in the bottom cluster start and end in different random poses. The target points on the most right side of each cluster represent the same position but due to their slightly different orientation, they are encoded on different latent points.

The results show that the method can successfully generate geodesics that respect the geometry of the manifold learned from demonstration. Interestingly, as shown by the magnification factor plot (Fig. 6-*left*), the resulting manifold is composed of two similar clusters, similarly to the illustrative example of Fig. 4. We observed that this behavior emerged due to the antipodal encoding of the quaternions, where each cluster represents one side of the hyper-sphere. It is worth highlighting that this encoding alleviates any kind of post-processing of raw quaternion data during training or reproduction phases.

Figure 6-*middle* shows one of the demonstrated trajectories on real robot, and Fig. 6-*right* displays the reconstructed geodesic using the decoder network and applied on a simulated robot arm. From the results, it is clear the motion generated by the geodesic leads to a motion pattern similar to the demonstrations. Note how the end-effector orientation evolves on the decoded geodesic in the ambient space, showing that the $90°$ rotation is properly encoded and reproduced.

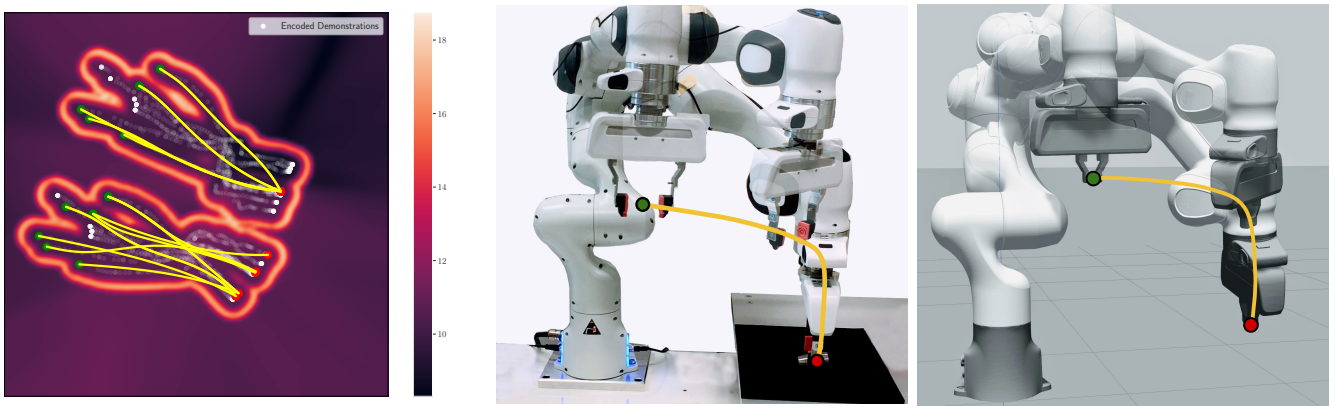

Fig. 6: *Left*: The yellow curves in the top cluster show geodesics starting from random points and ending up at the same target, and the curves in the bottom cluster connect random points on the manifold. The background depicts magnification factor derived from the learned manifold, and the semi-transparent white points show the encoded training dataset. *Middle*: It illustrates one of the demonstrations on the real robot, where the yellow curve depicts the end-effector trajectory. *Right*: It shows the reconstructed geodesic using the decoder network applied on the simulated robot. The yellow curve depicts the decoded geodesic computed on the learned manifold.

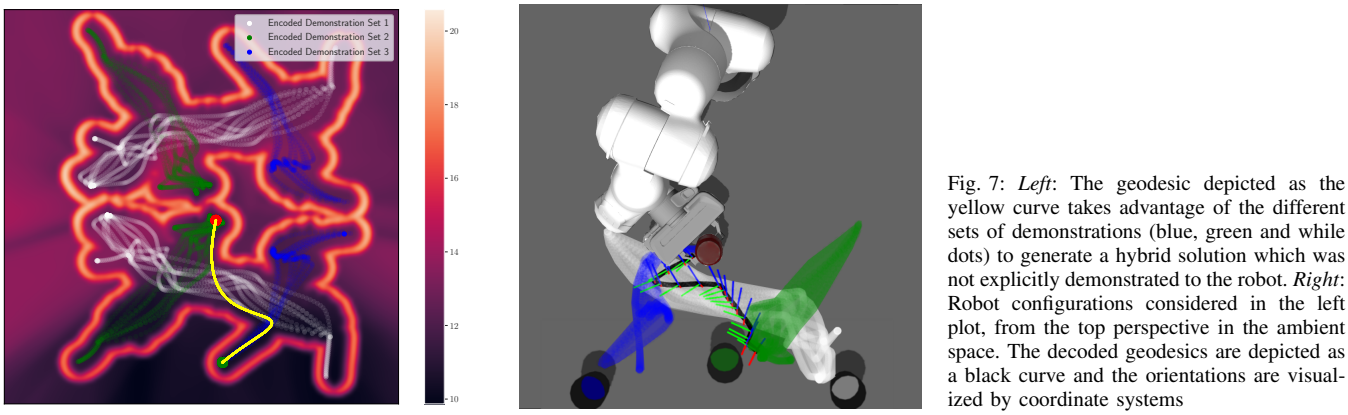

Fig. 7: *Left*: The geodesic depicted as the yellow curve takes advantage of the different sets of demonstrations (blue, green and while dots) to generate a hybrid solution which was not explicitly demonstrated to the robot. *Right*: Robot configurations considered in the left plot, from the top perspective in the ambient space. The decoded geodesics are depicted as a black curve and the orientations are visualized by coordinate systems

D. Simulated pouring task

To evaluate our model on a more complicated scenario, we collected a dataset of pouring task demonstrations on a simulator. The task involves grasping 3 cans from 3 different positions and then pouring at 3 different cups placed at different locations. The demonstrated trajectories cross each other, therefore providing a multiple-solution setting. Figure 7 shows how the geodesic, depicted as a yellow curve, is constructed in the latent space. This geodesic starts from a point on the second demonstration group (blue dots) and switches to the first group (white dots) to get to the given target located in the third group (green dots). As a result, with 3 sets of demonstrations, all 9 permutations for grasping any can from the table and then pouring any cup are feasible.

To evaluate the method in the presence of an unseen obstacle, the start and target points of the geodesic path are selected such that it follows one of the demonstration groups. To ensure that the obstacle is in the way, we select its position from the training set. Figure 8-*left* shows the geodesic performing obstacle avoidance while following the geometry of the manifold. The circular obstacle representation depicted as red and yellow circles in the latent space is just for the sake of visualization. The red and yellow curves represent geodesics avoiding the red and yellow obstacles, correspondingly. These curves correspond to one time frame of the adapted geodesics, showing how our method can deal with dynamic obstacles. The middle and right plots in Fig. 8 show the decoded geodesics executed on the simulated robot in the ambient space. The black trajectory shows the decoded geodesic, the dot clusters in red, green, and blue depict the demonstration sets, and the blue sphere depicts the obstacle. The middle plot shows the robot configuration 15 frames earlier than the plot at the right, displaying the obstacle dynamics during the task execution. Figure 9-*middle* illustrates the graph computed from the corresponding data manifold in Fig. 9-*left*. The graph-based geodesic (red curve in Fig. 9-*middle*) is then decoded and executed on the simulated robot (see Fig. 9-*right*). Although gradient-based and graph-based geodesic computation are both viable options, the latter is faster and thus more suitable for real-time motion generation.

VI. Discussion and Conclusions

We have proposed a novel LfD approach that learns a Riemannian manifold from human demonstrations and computes geodesics to recover and synthesize learned motion skills. Our proposed geodesic motion generation is capable of planning movements from and to arbitrary points on the data manifold,

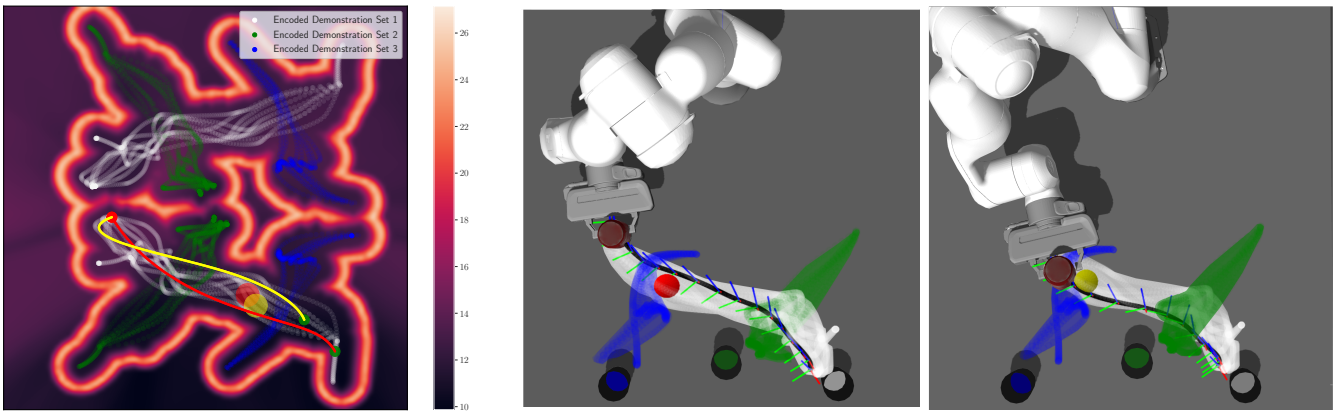

Fig. 8: *Left*: Red and yellow curves depict geodesics at two different time frames of the same motion, showing how the method computes paths that avoid dynamic obstacles (depicted as red and yellow circles). The background depicts magnification factor derived from the learned manifold. The dot clusters in red, green and blue depict the encoded demonstration sets. *Middle* and *right*: Robot configurations for the two time frames considered in the left plot, from the top perspective in the ambient space. The decoded geodesics are depicted as black curves and the orientations are visualized by coordinate systems.

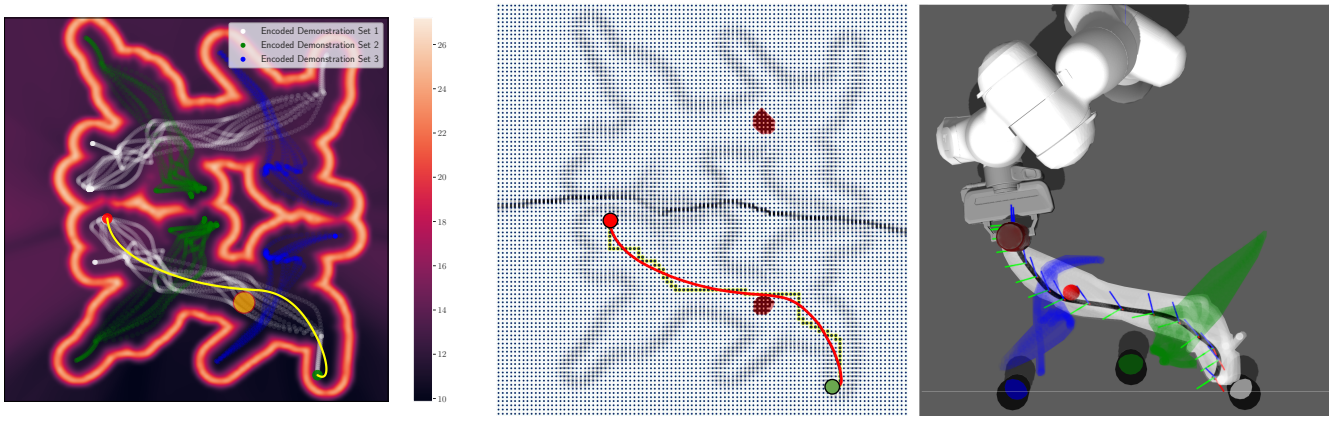

Fig. 9: *Left*: Geodesic computed on the graph is depicted as yellow curve avoiding the obstacle depicted as yellow circle. The background depicts magnification factor derived from the learned manifold. The dot clusters in red, green and blue depict the encoded demonstration sets. *Middle*: The graph computed from metric learned from data. *Right*: Robot configurations from the top perspective in the ambient space. The decoded geodesic is depicted as a black curve and the orientations are visualized by coordinate systems.

while avoiding obstacles on the fly. We realize the idea with a variational autoencoder (VAE) over the space of position and orientations of the robot end–effector. Motion is generated with graph–based geodesic computation for real–time motion generation and adaptation. Through extensive evaluation in the simulation, we show geodesic motion generation performs well in different scenarios such as grasping and pouring.

The proposed methodology can be extended and improved in several directions. A consequence of learning a manifold skill using VAEs is that data lying outside the manifold may be arbitrarily misrepresented in the latent space $\mathcal{Z}$. Consequently, any reconstruction in the ambient space $\mathcal{X}$ may be inaccurate. This may give rise to problems when conditioning on points, e.g. new targets, that are located outside the learned manifold. We did not explore this setting in the present paper. One possible solution may involve learning a bijective mapping between old demonstrations and new conditions, and then use this function to transform the learned manifold (e.g., by expanding or rotating) to fit another region of the space.

We introduce dynamical obstacle avoidance as a soft constraint through the ambient metric. If an approach based on hard constraints is to be preferred then one may opt to remove nodes near an obstacle from the graph instead of re-weighting edges. This could provide a computational saving, but one would lose the 'complete' Riemannian picture that we find elegant. It would also be straightforward to direct geodesics to go through select *via-points* by slight modifications to the graph algorithm. We did not explore these approaches here.

Our obstacle avoidance formulation only considered simple obstacles at this point, but the strategy can be extended to multiple dynamic obstacles. Instead of working with single Gaussian balls, one can imagine extending the approach to complex obstacle shapes represented as point clouds. This may increase the implementation demands in order to remain real time, but such an extension seems entirely reasonable. It is worth pointing out that to execute a motion safely the obstacle avoidance should be considered for all robot links and not just the end-effector. This requires a more informative manifold that is embedded in the joint space of the robot and an ambient space metric that combines the obstacle information in the

Euclidean and joint space simultaneously.

REFERENCES

[1] Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.11379.

[2] Georgios Arvanitidis, Søren Hauberg, Philipp Hennig, and Michael Schober. Fast and robust shortest paths on manifolds learned from data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1506–1515, 2019. URL http://proceedings.mlr.press/v89/arvanitidis19a.html.

[3] Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. URL https://arxiv.org/abs/2008.00565.

[4] Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5058–5069, 2020. URL https://proceedings.neurips.cc/paper/2020/file/354ac345fd8c6d7ef634d9a8e3d47b83-Paper.pdf.

[5] Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Magnification factors for the SOM and GTM algorithms. In *Workshop on Self-Organizing Maps*, 1997. URL https://research.aston.ac.uk/files/113377/NCRG_97_008.pdf.

[6] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016. URL https://doi.org/10.1007/s11370-015-0187-9.

[7] Ricardo Campa and Karla Camarillo. Unit quaternions: A mathematical tool for modeling, path planning and control of robot manipulators. In *Robot Manipulators*, chapter 2. IntechOpen, Rijeka, 2008. URL https://doi.org/10.5772/6197.

[8] Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1550, 2018. URL http://proceedings.mlr.press/v84/chen18e.html.

[9] Nutan Chen, Francesco Ferroni, Alexej Klushyn, Alexandros Paraschos, Justin Bayer, and Patrick van der Smagt. Fast approximate geodesics for deep generative models. In *International Conference on Artificial Neural Networks (ICANN)*, pages 554–566, 2019. URL https://link.springer.com/content/pdf/10.1007%2F978-3-030-30484-3_45.pdf.

[10] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[11] Piotr Dollár, Vincent Rabaud, and Serge Belongie. Non-isometric manifold learning: Analysis and an algorithm. In *International Conference on Machine Learning (ICML)*, pages 241–248, 2007. doi: 10.1145/1273496.1273527.

[12] David Eklund and Søren Hauberg. Expected path length on random manifolds. In *arXiv preprint*, 2019. URL https://arxiv.org/abs/1908.07377.

[13] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014. doi: 10.1109/ACCESS.2014.2302442.

[14] Marco Ewerton, Gerhard Neumann, Rudolf Lioutikov, Heni Ben Amor, Jan Peters, and Guilherme Maeda. Learning multiple collaborative tasks with a mixture of interaction primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1535–1542, 2015. URL https://doi.org/10.1109/ICRA.2015.7139393.

[15] Søren Hauberg. Only Bayes should learn a manifold. 2019.

[16] Søren Hauberg, Aasa Feragen, Raffi Enficiaud, and Michael J. Black. Scalable robust principal component analysis using Grassmann averages. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(11):2298–2311, 2016. URL https://doi.org/10.1109/TPAMI.2015.2511743.

[17] Ioannis Havoutis and Subramanian Ramamoorthy. Motion planning and reactive control on learnt skill manifolds. *International Journal of Robotics Research (IJRR)*, 32(9-10):1120–1150, 2013. doi: 10.1177/0278364913482016. URL https://doi.org/10.1177/0278364913482016.

[18] Evan G. Hemingway and Oliver M. O'Reilly. Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. *Multibody System Dynamics*, 44(1):31–56, mar 2018. doi: 10.1007/s11044-018-9620-0. URL https://doi.org/10.1007%2Fs11044-018-9620-0.

[19] Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G Caldwell. Kernelized movement primitives. *International Journal of Robotics Research (IJRR)*, 38(7):833–852, 2019. URL https://doi.org/10.1177/0278364919846363.

[20] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25:328–373, 2013.

[21] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets Riemannian manifolds in robot learning. In *Proceedings of the Conference on Robot Learning*, pages 233–246, 2020. URL http://proceedings.mlr.press/v100/jaquier20a.html.

[22] Dimitris Kalatzis, David Eklund, Georgios Arvanitidis, and Søren Hauberg. Variational autoencoders with Riemannian Brownian motion priors. In *International Conference on Machine Learning (ICML)*, volume 119, pages 6789–6799, 2020. URL http://proceedings.mlr.

press/v119/kalatzis20a/kalatzis20a.pdf.

[23] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014. URL https://arxiv.org/abs/1312.6114.

[24] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research (IJRR)*, 31(3):360–375, 2012. URL https://doi.org/10.1177/0278364911428653.

[25] John Lee. *Introduction to Riemannian Manifolds*. Springer, 2 edition, 2018.

[26] Mao Li, Kenji Tahara, and Aude Billard. Learning task manifolds for constrained object manipulation. *Autonomous Robots*, 42:159–174, 2018. doi: https://doi.org/10.1007/s10514-017-9643-z.

[27] M.G. Mohanan and Ambuja Salgoankar. A survey of robotic motion planning in dynamic environments. *Robotics and Autonomous Systems*, 100:171–185, 2018. ISSN 0921-8890. URL https://www.sciencedirect.com/science/article/pii/S0921889017300313.

[28] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1–2):1–179, 2018. ISSN 1935-8253. doi: 10.1561/2300000053.

[29] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42:529–551, 2018.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035. 2019.

[31] Nathan Ratliff, Matthew Zucker, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 489–494, 2009. URL https://doi.org/10.1109/ROBOT.2009.5152817.

[32] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020. URL https://doi.org/10.1146/annurev-control-100819-063206.

[33] Leonel Rozo, Pablo Jimenez, and Carme Torras. Robot learning from demonstration of force-based tasks with multiple solution trajectories. In *International Conference on Advanced Robotics (ICAR)*, pages 124–129, 2011. URL https://doi.org/10.1109/ICAR.2011.6088633.

[34] Leonel Rozo, Meng Guo, Andras G. Kupcsik, Marco Todescato, Philipp Schillinger, Markus Giftthaler, Matthias Ochs, Markus Spies, Nicolai Waniek, Patrick Kesper, and Mathias 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. URL https://doi.org/10.1109/IROS45743.2020.9341570.

[35] Aidan Scannell, Carl Henrik Ek, and Arthur Richards. Trajectory Optimisation in Learned Multimodal Dynamical Systems Via Latent-ODE Collocation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[36] Stefan Schaal. Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics. In *Adaptive Motion of Animals and Machines*, pages 261–280. Springer-Verlag, Tokyo. doi: 10.1007/4-431-31381-8_23. URL http://link.springer.com/10.1007/4-431-31381-8{_}23.

[37] M. Yunus Seker, Mert Imre, Justus H. Piater, and Emre Ugur. Conditional neural movement primitives. In *Robotics: Science and Systems (R:SS)*, 2019.

[38] Hang Shao, Abhishek Kumar, and P. Thomas Fletcher. The Riemannian geometry of deep generative models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 428–4288, 2018. URL https://doi.org/10.1109/CVPRW.2018.00071.

[39] Saif Sidhik. panda_simulator: Gazebo simulator for Franka Emika Panda robot supporting sim-to-real code transfer, 2020. URL https://doi.org/10.5281/zenodo.3747459.

[40] Saif Sidhik. justagist/franka_ros_interface: Franka ROS Interface version for franka-ros v0.7.1-release, 2020. URL https://doi.org/10.5281/zenodo.4320612.

[41] Suvrit Sra. Directional statistics in machine learning: A brief review. In Christophe Ley and Thomas Verdebout, editors, *Applied Directional Statistics*. 2018. 1st edition.