Reactive Motion Generation on Learned Riemannian Manifolds

Hadi Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, Gerhard Neumann, and Leonel Rozo

Abstract
In recent decades, advancements in motion learning have enabled robots to acquire new skills and adapt to unseen conditions in both structured and unstructured environments. In practice, motion learning methods capture relevant patterns and adjust them to new conditions such as dynamic obstacle avoidance or variable targets. In this paper, we investigate the robot motion learning paradigm from a Riemannian manifold perspective. We argue that Riemannian manifolds may be learned via human demonstrations in which geodesics are natural motion skills. The geodesics are generated using a learned Riemannian metric produced by our novel variational autoencoder (VAE), which is especially intended to recover full-pose end-effector states and joint space configurations. In addition, we propose a technique for facilitating on-the-fly end-effector/multiple-limb obstacle avoidance by reshaping the learned manifold using an obstacle-aware ambient metric. The motion generated using these geodesics may naturally result in multiple-solution tasks that have not been explicitly demonstrated previously. We extensively tested our approach in task space and joint space scenarios using a 7-DoF robotic manipulator. We demonstrate that our method is capable of learning and generating motion skills based on complicated motion patterns demonstrated by a human operator. Additionally, we assess several obstacle avoidance strategies and generate trajectories in multiple-mode settings.

Keywords
Robot Motion Learning, Manifold Learning, Riemannian Manifolds, Geodesic Motion Skills

1 Introduction
Robot learning has gained interest recently because of its potential to endow robots with a repertoire of motion skills to operate autonomously or assist humans in repetitive, precise, and risk-averse tasks. In this paper, we tackle the robot motion generation problem from a robot learning perspective, which involves modeling and reproducing motion skills such as picking, placing, etc. In practice, we may broadly classify motion generation techniques into two categories: Classic motion planners and movement primitives. The former generates obstacle-free continuous trajectories between start and goal configurations using search or sampling-based algorithms (Elbanhawi and Simic 2014; Mohanan and Salgoankar 2018). Movement primitives, in contrast, are a modular abstraction of robot movements, where a primitive represents an “atomic action” or an “elementary movement”, which are often designed using robot learning techniques (Schaal et al. 2003). A relevant distinction between the aforementioned approaches is that motion planners require a precise description of the robot workspace to generate a continuous path to the goal, while movement primitives rely on observed motion patterns to encapsulate the desired trajectory without an explicit model of the environment. If an unseen obstacle shows up in the robot workspace, motion planners often need to replan the entire trajectory from scratch, while most movement primitives require to be retrained, unless they are provided with via-point passing features. We aim for a trade-off between these two categories: A learning-based reactive motion generation that generates robot movements resembling the observed motion patterns while avoiding obstacles on the fly.

We resort to learning-from-demonstration (LfD) to build a model of robot motions, a technique in which the skill model is learned by encapsulating motion patterns from human demonstrations (Osa et al. 2018). The current LfD approaches do not need an environment model and some can quickly adapt to settings with dynamic targets (Osa et al. 2018; Ijspeert et al. 2013; Calinon 2016). We can broadly identify three main groups of LfD approaches that use motion primitives as their building blocks. The first group explicitly considers the motion dynamics (Ijspeert et al. 2013; Manschitz et al. 2018). The second group builds on probabilistic approaches that exploit demonstration variability to build a model of the environment using reinforcement learning and planning (CR/PJ-AI-R31), 70049 Stuttgart, GERMANY

Corresponding author:
Hadi Beik-Mohammadi, Bosch Center for Artificial Intelligence (BCAI), Reinforcement learning and planning (CR/PJ-AI-R31), 70049 Stuttgart, GERMANY
Email: hadi.beik-mohammadi@de.bosch.com
The last group uses neural networks as their skill model and focuses on robot motion generalization (Seker et al. 2019; Osa and Ikemoto 2019). Despite their significant contributions (see Section 2), several challenges are still open: Encoding and reproducing full-pose end-effector movements, skill adaptation to unseen or dynamic obstacles at both task and joint spaces, handling multiple-solution (a.k.a. multiple-mode) tasks in the aforementioned spaces, and generalization to unseen situations (e.g. new target poses), among others.

In our previous paper (Beik-Mohammadi et al. 2021), we provided an LfD approach that addressed several of foregoing problems by leveraging a Riemannian perspective for learning robot motions using task space demonstrations. We employed a Riemannian formulation to represent a motion skill, in which human demonstrations were assumed to form a Riemannian manifold (i.e. a smooth surface), which could be learned in task space $\mathbb{R}^3 \times S^3$. This approach showed to have a number of advantages: Strong capabilities to model the non-linearity of motion data, adaptability to unseen conditions (e.g. obstacles), and generation of novel task solutions. The core of our previous approach built on a variational autoencoder (VAE) that was exploited to compute a Riemannian metric describing the underlying structure of these skill manifolds. This metric allowed us to measure distances on Riemannian manifolds, a property that is used to compute geodesics (i.e., the shortest paths on Riemannian manifolds), which was then leveraged as our motion generation mechanism. In addition, we designed an ambient space metric to reshape the skill manifold on the fly in order to perform obstacle avoidance. Furthermore, the learned skill manifold was able to encode multiple-solution tasks, which naturally resulted in novel hybrid solutions based on the synergy of a subset of the demonstrations.

In this paper, we extend our previous work in two different ways. First, inspired by the need of avoiding obstacles at any location of the robot body, we propose a new reactive motion generation method that also leverages the Riemannian approach proposed in (Beik-Mohammadi et al. 2021) for joint space skills. To do so, we develop a new VAE architecture that integrates the robot forward kinematics to access task space information of any point on the robot body. This new approach makes joint space skill adaptation possible, allowing the robot to simultaneously avoid unseen and dynamic obstacles and handle multiple-solution (a.k.a. multiple-mode) tasks in both task and joint spaces. Second, we describe both methodologies with simple examples and extensively evaluate them on the real tasks using a 7-DoF robot manipulator. The experiments showcase how our techniques allow a robot to learn and reconstruct motion skills of varying complexity. In particular, we show how our methods enable obstacle avoidance at task and joint space levels. Moreover, we provide experiments where the robot exploits multiple-solution tasks to effectively generate hybrid solutions without model retraining. Furthermore, we experimentally analyze how the latent space dimensionality impacts the quality of the geodesic motion generator, and how the robot behaves when controlled by geodesics crossing the manifold boundaries.

In summary, we propose a new Riemannian learning framework for reactive motion skills that lies on the middle ground between classical motion planning and motion primitives. Similar to motion primitives, we exploit human demonstrations to learn a model of a skill with the assumption that these demonstrations belong to a smooth surface characterized as a Riemannian manifold. Additionally, our method, like motion planners, derives a time-independent reference trajectory from a geodesic motion generator while incorporating obstacle avoidance capabilities.

In Section 2, we review relevant work on learning from demonstration, variational auto-encoders, Riemannian manifolds, their connections, and ambient space metrics. In Section 3, we examine how a skill manifold may be trained and used to reconstruct robot motion when the demonstrations are provided in task and joint spaces. Section 4 investigates geodesic computation and their ability to avoid unseen obstacles on the fly. In Section 5, we show our method capabilities via a series of real-world robot experiments. Finally, in Section 6, we discuss some of the limitations of our approach and provide possible solutions to be explore in future work.

2 Background and Related Work

Due to the strong connections between our work and the learning from demonstration (LfD) technique, we begin this section by discussing pertinent work on learning and synthesizing joint and task space motion skills via LfD. We look into how these approaches reconstruct full-pose trajectories, provide obstacle avoidance capabilities, and reproduce multiple-solution (multiple-mode) tasks. Second, we review Riemannian geometry as well as variational auto-encoders (VAEs), which are essential concepts for our approach. Finally, we explain recent advances in the connection between VAEs and Riemannian geometry, which is the backbone of our methods.
2.1 Learning from Demonstration (LfD)

LfD is a robot programming technique that leverages human demonstrations, recorded via kinesthetic teaching or teleoperation, to learn a model of a task (Ravichandar et al. 2020). End-effector positions and orientations, joint configurations, linear or angular velocities, and accelerations are all examples of data that can be used in LfD. The methods for dealing with motion dynamics are outside the scope of this study; instead, we focus on techniques for learning and synthesizing robot skills built on joint and task space trajectories.

We identify three key lines of research and their particular features (e.g. obstacle avoidance) for robot motion learning. The first group focuses on motion dynamics learning (Ijspeert et al. 2013; Manschitz et al. 2018), where demonstrations are considered as solutions to specific dynamical systems. These techniques are well behaved when confronted with changes in the environment due to their autonomous-systems formulation. The second set of approaches builds on probabilistic methods that exploit data variability and model uncertainty (Huang et al. 2019; Calinon 2016; Paraschos et al. 2018). Their probabilistic formulation allows robots to execute the skill using a large diversity of trajectories sampled from the skill model. The last category includes approaches that use neural networks for increasing generalization in robot motion learning (Seker et al. 2019; Osa and Ikemoto 2019). Furthermore, there exist methods that combine dynamical systems and neural networks (Bahl et al. 2020; Rana et al. 2020), or dynamical systems and probabilistic models (Ugur and Girgin 2020; Khansari-Zadeh and Billard 2011). All of the aforementioned methods belong to the category of movement primitives (MPs), which is an alternative approach to classic motion planning (Elbanhawi and Simic 2014) for robot motion generation.

In contrast, we propose a reactive motion generation technique that lies on a middle ground between movement primitives and motion planning. Similar to motion primitives, we exploit human demonstrations to learn a model of a skill with the assumption that these demonstrations belong to a smooth surface characterized as a Riemannian manifold. Additionally, our method, like motion planners, derives a time-independent reference trajectory generated from geodesic curves, which can be locally deformed to avoid unseen obstacles. Specifically, 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 via geodesics. The decoded geodesics are then decoded and used as reference trajectories on the robot to reproduce motion trajectories that resemble the demonstrations.

Complex robot movements may involve sophisticated full-pose end-effector trajectories, making it imperative to have a learning framework capable of encoding full-pose motion patterns. The main challenge is then how to properly encode and reproduce orientation movements. Despite most LfD approaches have overlooked this problem (Paraschos et al. 2018; Seker et al. 2019; Calinon 2016; Huang et al. 2019), recent works have addressed it using probabilistic models (Rozo and Dave 2021; Zeestraten 2018). In our previous work (Beik-Mohammadi et al. 2021), we proposed a VAE architecture capable of encoding full-pose trajectories, which is here exploited for learning a variety of real robotic tasks.

Obstacle avoidance is another feature that a reactive motion generation should offer. While several approaches rely solely on obstacles information given before learning (Prescott and Mayhew 1992; Aljalbout et al. 2020), these are ineffective in dynamic environments. Other techniques exploit via-points (Paraschos et al. 2018; Seker et al. 2019; Huang et al. 2019) to tackle this problem in an indirect fashion, but they do not require retraining the skill model. A different perspective to the obstacle avoidance problem is to see obstacle as costs in an optimization framework that seeks to generate optimal and obstacle-free trajectories (Urain et al. 2021; Ratliff et al. 2018).

Meanwhile, our method tackles the obstacle avoidance problem from a metric reshaping viewpoint. We design obstacle-aware ambient space metrics to reshape the learned Riemannian metric. The combination of these two metrics yields a new metric that is exploited to generate trajectories that simultaneously follow the demonstrations while also avoiding obstacles. Note that the ambient space metric always represents a notion of distance to the obstacles in task space. It is worth mentioning that the choice of demonstration ambient space defines how this metric can be designed to avoid obstacles. For example, when using joint space demonstrations, the ambient space metric can incorporate information about the distance from any point of the robot to the obstacles, resulting in a multiple-limb obstacle avoidance capability. On the contrary, when using task space end-effector demonstrations, this metric only provides obstacle avoidance at the level of the robot end-effector.

We believe our obstacle avoidance technique is conceptually comparable to CHOMP (Ratliff et al. 2009) since it also uses a simplified geometric description of the robot to construct a uniform grid in task space to determine whether a trajectory may cause collision with an obstacle.

Human demonstrations can naturally show various solutions to a single motion skill (Rozo et al. 2011; Seker et al. 2019), which is typically addressed using hierarchical techniques (Konidaris et al. 2012; Ewerton et al. 2015). Our approach enables the encoding of multiple-solution tasks on the learned Riemannian manifold, which is then exploited to replicate the demonstrated skill and generate novel hybrid solutions based on a synergy of a (sub)set of the demonstrations. These hybrid solutions naturally emerge from our geodesic motion generator on the learned Riemannian manifold. Note that the aforementioned robot motion learning techniques only provide trajectories that strictly follow the demonstrated patterns without the ability to generate hybrid solutions.

2.2 Variational auto-encoders (VAEs)

VAEs are generative models (Kingma and Welling 2014) that learn and reconstruct data by encapsulating their density into a lower-dimensional latent space $z$. Specifically, VAEs encode the training data density $p(x)$ in an ambient space $X$ through a low-dimensional latent variable $z$. For simplicity, we consider the generative process of a Gaussian VAE

$$p_{\theta}(x) = \mathcal{N}(x | \mu, \Sigma)$$

where $\mu$ and $\Sigma$ are the mean and covariance of the normal distribution, respectively. The encoder $q_{\phi}(z|x)$ is a variational distribution that approximates the true posterior $p(z|x)$. It is typically a Gaussian distribution with mean $\mu_q$ and covariance $\Sigma_q$.

$$q_{\phi}(z|x) = \mathcal{N}(z | \mu_q, \Sigma_q)$$

The decoder $p_{\theta}(x|z)$ is a conditional distribution that maps the latent variable $z$ to the data space $X$. It is usually a feedforward neural network that takes $z$ as input and outputs the probability distribution of $x$.

$$p_{\theta}(x|z) = \mathcal{N}(x | \mu_{\theta}(z), \Sigma_{\theta}(z))$$

where $\mu_{\theta}(z)$ and $\Sigma_{\theta}(z)$ are the mean and covariance of the normal distribution, respectively. The generator $p_{\theta}(x)$ is the overall probability distribution of the data space $X$, which is the product of the marginal likelihood $p_{\theta}(x)$ and the prior $q_{\phi}(z)$.

$$p_{\theta}(x) = \int q_{\phi}(z) p_{\theta}(x|z) dz$$

The VAE is trained by minimizing the following objective function:

$$\mathbb{E}_{q_{\phi}(z|x)} \left[ \log p_{\theta}(x|z) - D_{KL}(q_{\phi}(z|x) \parallel p_{\theta}(z)) \right]$$

where $D_{KL}$ is the Kullback-Leibler divergence between $q_{\phi}(z|x)$ and $p_{\theta}(z)$. The first term in the objective function is the reconstruction error, which measures the difference between the original data $x$ and the reconstructed data $\tilde{x} = \mathbb{E}_{q_{\phi}(z|x)} [p_{\theta}(x|z)]$. The second term is the regularization term, which pushes the latent distribution $q_{\phi}(z|x)$ to be close to the prior distribution $p_{\theta}(z)$.
In differential geometry, Riemannian manifolds are referred to as curved manifolds \( M \). To learn a skill-specific Riemannian manifold from human data, a distribution of the manifold, which locally resembles the Euclidean space, can be computed given the curve length in Eq. (6). Geodesics on Riemannian manifolds can be seen as a generalization of straight lines in Euclidean space. However, geodesics might not be unique, e.g., great circles on the sphere manifold. Later, we demonstrate that calculating geodesics on a learned Riemannian manifold can be leveraged to recover demonstrated motion patterns. It should be noted that geodesics have recently been utilized as solutions of trajectory optimizers for quadrotor control (Scannell et al. 2021).

### 2.4 Learning Riemannian Manifolds with VAEs

In this subsection, we examine the link between VAEs and Riemannian geometry. To begin, we first define the VAE generative process of Eq. (2) as a stochastic function,

\[
    f_\phi(z) = \mu_\phi(z) + \sigma_\phi(z) \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbb{I}_D). \tag{8}
\]

where \( \mu_\phi(z) \) and \( \sigma_\phi(z) \) are decoder mean and variance neural networks, respectively. Also, \( \text{diag}(\cdot) \) is a diagonal matrix, and \( \mathbb{I}_D \) is a \( D \times D \) identity matrix. The above formulation is referred to as the reparameterization trick (Kingma and Welling 2014), which can be interpreted as samples generated out of a random projection of a manifold jointly spanned by \( \mu_\phi \) and \( \sigma_\phi \), as depicted in Fig. 2.

Riemannian manifolds may arise from mapping functions between two spaces as in Eq. (4). As a result, Eq. (8) may be seen as a stochastic version of the mapping function of Eq. (4), which in turn defines a Riemannian manifold (Hauberg 2019). We can now write the stochastic form of the Riemannian metric of Eq. (7). To do so, we first recast the stochastic function Eq. (8) as follows (Eklund and Hauberg 2019),

\[
    f_\phi(z) = (\mathbb{I}_D, \text{diag}(\epsilon)) \begin{pmatrix} \mu_\phi(z) \\ \sigma_\phi(z) \end{pmatrix} = P g(z), \tag{9}
\]

where \( P \) is a random matrix, and \( g(z) \) is the concatenation of \( \mu_\phi(z) \) and \( \sigma_\phi(z) \). Therefore, the VAE can be seen as a random projection of a deterministic manifold spanned by \( g(z) \).

Given this stochastic mapping function is defined by a combination of mean \( \mu_\phi(z) \) and variance \( \sigma_\phi(z) \), the metric is likewise based on a mixture of both as follows,

\[
    M(z) = J_{\mu_\phi}(z)^T J_{\mu_\phi}(z) + J_{\sigma_\phi}(z)^T J_{\sigma_\phi}(z). \tag{10}
\]

where \( J_{\mu_\phi}(z) \) and \( J_{\sigma_\phi}(z) \) are respectively the Jacobian of \( \mu_\phi(z) \) and \( \sigma_\phi(z) \) evaluated at \( z \in \mathbb{Z} \), with \( \mathbb{Z} \) being the VAE low-dimensional latent space.

Notably, the decoder variance network \( \sigma_\phi(z) \) approximates the data uncertainty, which plays a critical role in the metric Eq. (10) by associating low values to regions with a high number of data points and vice-versa. Indeed, omitting this element results in a nearly flat manifold geometry (Hauberg 2019). For example, Shao et al. (2018) suggest a similar technique that learns Riemannian manifolds using generative models that do not model data uncertainty, resulting in flat manifolds and often straight lines as geodesics. As explained earlier, the Riemannian metric is required to compute the geodesics, which conform to the geometry of the training data (Arvanitidis et al. 2018).

In summary, we exploit the link between VAEs and Riemannian metrics for robot motion generation. Specifically, we learn a Riemannian metric that describes the motion patterns observed during the demonstrations. These demonstrations may take place in two different
ambient spaces: Task and joint space, which define the VAE architecture needed to learn the manifold of interest. The geodesic curves generated on the learned manifold produce robot movements mimic the given demonstrations in the ambient space.

3.2.1 Position encoding in $\mathbb{R}^3 \times S^3$ To begin, we focus on learning motion skills characterized by full-pose end-effector trajectories, where each pose is represented in $\mathbb{R}^3 \times S^3$. Before exploiting the VAE to compute the Riemannian metric, we must ensure its capability to properly learn and reconstruct full-pose end-effector states, i.e. position $x \in \mathbb{R}^3$ and orientation $q \in S^3$, while accounting for specific properties of the data, such as quaternions antipodality. To do so, we propose a VAE architecture that models the joint density of the robot end-effector state. Our model retains the usual Gaussian prior $p(z) = N(z|0, I_d)$, but modifies the generative distribution $p_{\phi, \psi}(x, q|z)$. Specifically, we assume that position and orientation are conditionally independent,

$$p_{\phi, \psi}(x, q|z) = p_{\phi}(x|z)p_{\psi}(q|z).$$

where the latent variable $z$ captures the correlation between position and quaternion data. Next, we describe how each conditional distribution is parameterized and learned.

3.1.1 Position encoding in $\mathbb{R}^3$: To model the conditional distribution of end-effector positions $x$, we opt for simplicity and choose this to be Gaussian,

$$p_{\phi}(x|z) = N(x|\mu_\phi(z), I_3\sigma_\phi^2(z)), \quad \text{(14)}$$

where $\mu_\phi$ and $\sigma_\phi$ are neural networks parametrized by $\phi$. One could argue that $p_{\phi}(x|z)$ should have zero probability mass outside the workspace of the robot, but we disregard such concerns as $\sigma_\phi^2$ 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.
3.1.2 Quaternion encoding in $S^3$: On a robot motion trajectory, each position is paired with an orientation, and together they define the full pose of the end-effector. There are several representations for the end-effector orientation, for example, Euler angles, rotation matrices, and unit quaternions. Euler angles and rotation matrices are widely used for their simplicity and intuitiveness, however, the former suffer from gimbal lock (Hemingway and O’Reilly 2018) which makes them an inadequate orientation parametrization, and the latter are a redundant representation requiring a high number of parameters.

Unit quaternions, on the other hand, are a convenient way to represent orientations since they are compact, not redundant, and prevent gimbal lock. Also, they provide strong stability guarantees in closed-loop orientation control (Campa and Camarillo 2008), and they have been recently exploited in complicated robotic tasks learning (Rozo et al. 2020), and for data-efficient robot control tuning (Jaquier et al. 2020) using Riemannian-manifold formulations. We choose to represent orientations $q$ as a unit quaternion, such that $q \in S^3$ with the additional antipodal identification that $q$ and $-q$ correspond to the same orientation. Formally, a unit quaternion $q$ lying on the surface of a 3-sphere $S^3$ can be represented using a 4-dimensional unit vector $q = [q_w, q_x, q_y, q_z]$, 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 model $q$ as a point in a projective space, but for simplicity we let $q$ live on the unit sphere $S^3$. We then choose a generative distribution $p_\psi(q|x)$ such that $p_\psi(q|z) = p_\psi(-q|z)$. In other words, the quaternions $q$ and $-q$ are considered to be antipodal: they lie on diametrically opposite points on the 3-sphere while representing the same orientation.

To formulate a suitable distribution $p_\psi(q|x)$ over $S^3$, we leverage the von Mises-Fisher (vMF) distribution, which is merely an isotropic Gaussian constrained to lie on the unit sphere (Sra 2018). This distribution is described by a mean direction $\mu$ with $||\mu|| = 1$, and a concentration parameter $\kappa \geq 0$. The vMF density function is defined as,

$$vMF(q|\mu, \kappa) = C_D(\kappa) \exp \left( \kappa \mu^T q \right), \quad ||\mu|| = 1, \quad (15)$$

where $C_D$ is the normalization constant

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

with $I_{\frac{D}{2}}(\kappa)$ being the modified Bessel function of the first kind. Like the Gaussian, from which the distribution was constructed, the von Mises-Fisher distribution is unimodal. To build a distribution that is antipodal symmetric, i.e. $p_\psi(q|x) = p_\psi(-q|x)$, we define a mixture of antipodal vMF distributions (Hauberg et al. 2016),

$$p_\psi(q|x) = \frac{1}{2} vMF(q|\mu_\psi(x), \kappa_\psi(x)) + \frac{1}{2} vMF(q|{-\mu_\psi(x)}, \kappa_\psi(x)), \quad (17)$$

where $\mu$ and $\kappa$ are parametrized as neural networks. This mixture model is conceptually similar to a Bingham distribution (Sra 2018), but is easier to implement numerically.

3.1.3 Variational inference: To train the VAE, we maximize an adapted evidence lower bound (ELBO) Eq. (3), defined as

$$L_{ELBO} = \beta_1 L_a + \beta_2 L_q - KL(q(z|x)||p(z)), \quad (18)$$

$$L_a = \mathbb{E}_{q(z|x)} [\log p_\phi(x|z)], \quad (19)$$

$$L_q = \mathbb{E}_{q(z|x)} [\log p_\psi(q|z)], \quad (20)$$

where $x \in \mathbb{R}^3$ and $q \in S^3$ represent the position and orientation of the end-effector, respectively. The scaling factors $\beta_1 > 0$ and $\beta_2 > 0$ balance the log-likelihood of position and orientation components. Due to the quaternions antipodality, 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 vMF distributions that encode the same orientation at both sides of the hypersphere. Practically, we double the training data, by including $q_n$ and $-q_n$ for all observations $q_n$. Note that as the Riemannian manifold is learned using task space data, the model is kinematics agnostic, which means that the generated motion may be used across different robots as long as the trajectory is reachable.

3.1.4 Induced Riemannian metric: Our generative process is parametrized by a set of neural networks. Specifically, $\mu_\phi$ and $\sigma_\phi$ are position mean and variance neural networks parameterized by $\psi$, while $\mu_\sigma$ and $\kappa_\sigma$ are neural networks parameterized by $\psi$ that represent the mean and concentration of the quaternion distribution. Following Section 2.4, the Jacobians of these functions govern the induced Riemannian metric as,

$$M(z) = M^\mu_\phi(z) + M^\sigma_\phi(z) + M^\mu_\sigma(z) + M^\kappa_\sigma(z), \quad (21)$$

with

$$M^\mu_\phi(z) = J_{\mu_\phi}(z)^T J_{\mu_\phi}(z), \quad (22)$$

$$M^\sigma_\phi(z) = J_{\sigma_\phi}(z)^T J_{\sigma_\phi}(z), \quad (23)$$

$$M^\mu_\sigma(z) = J_{\mu_\sigma}(z)^T J_{\mu_\sigma}(z), \quad (24)$$

$$M^\kappa_\sigma(z) = J_{\kappa_\sigma}(z)^T J_{\kappa_\sigma}(z), \quad (25)$$

where $J_{\mu_\phi}, J_{\sigma_\phi}, J_{\mu_\sigma}, J_{\kappa_\sigma}$ are the Jacobians of the functions representing the position mean and variance, as well as the quaternion mean and concentration, respectively.

In practice, we want this Riemannian metric $M(z)$ to take large values in regions with little or no data, so that geodesics avoid passing through them. We achieve this by using radial basis function (RBF) networks as our variance representation, whose kernels reliably extrapolate over the whole space (Arvanitidis et al. 2018). Since one of the main differences between Gaussian and von Mises-Fisher distributions is the representation of the 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 $\sigma_\phi^{-1}(z)$ and $\kappa_\sigma(z)$, which affect the Riemannian metric through their corresponding Jacobians as in Eq. (21).
3.2 Joint space $\mathbb{R}^\eta$

The joint space $\mathbb{R}^\eta$, also known as configuration space, is another space to represent robot motion trajectories. In this space, each trajectory point is represented as the vector $\theta = [\theta_1, \theta_2, \ldots, \theta_n] \in \mathbb{R}^\eta$, where $\eta$ is the number of degrees of freedom of the robot. Learning motion skills in this space is known to be challenging as it is less intuitive to provide joint-level demonstrations. However, being able to learn and generate joint space movements is relevant as some tasks may demand specific robot postures. Moreover, joint space skills can be extended to provide whole-body obstacle avoidance. In this context, we formulate a Riemannian robot motion learning approach to generate collision-free joint space movements.

Previously, we computed a Riemannian metric in the latent space using the VAE decoder trained on task space demonstrations. Similarly, a new VAE architecture can be designed to compute a Riemannian metric from joint space demonstrations. We use this metric to compute geodesics and generate robot movements that resemble the demonstrations in joint space. This joint space approach also allows us to endow the robot with whole-body obstacle avoidance capabilities. By using ambient metrics, we can again reshape the learned metric to make the robot move away from obstacles in an online fashion. The ambient metrics exclusively uses task space information of the obstacles and the robot body, in contrast to classical motion planning that often works in the configuration space. Note that the data manifold learned using joint space demonstrations is kinematics-dependent, meaning that the generated motion cannot be directly transferred to other robots with different kinematics.

3.2.1 Variational inference: To train the joint space VAE, we maximize a modified version of the evidence lower bound (ELBO) Eq. (3), defined as

$$
\mathcal{L}_{ELBO} = \mathcal{L}_\theta - KL (q(z_i|\theta_i)||p(z)) ,
$$

(26)

with,

$$
\mathcal{L}_\theta = \mathbb{E}_{q(z_i|\theta_i)} [p_X(FK(\theta)|z_i)],
$$

$$
= \mathbb{E}_{q(z_i|\theta_i)} [\log(p_\theta(\theta|z_i)) - \log(\mathcal{V})],
$$

where $p_\theta(\theta|z_i)$ and $p_X(FK(\theta)|z_i)$ are the estimated conditional densities in the joint space $\Theta$ and task space $X$, respectively. Also, $\mathcal{V}$ is the volume measure defined as

$$
\mathcal{V} = \sqrt{\det(J^T_{FK}(\mu_{\alpha}^T(z)), J_{FK}(\mu_{\alpha}(z)))},
$$

(27)

where $J_{FK}$ is the Jacobian of the forward kinematics $FK$ given the joint configuration estimated by the VAE decoder $\mu_{\alpha}$. Furthermore, the generative distribution $p_\theta(\theta|z_i) = N(\mu_{\alpha}(z_i), \Sigma_\alpha(z_i)^2)$ is parameterized by the VAE decoder mean $\mu_{\alpha}(z)$ and variance $\sigma_\alpha(z)$ networks. Note that the new ELBO formulation in Eq. (26) leverages the change of variable theorem (Deisenroth et al. 2020) to transform probability densities from joint to task space. As a result, the VAE is still trained using task space information, while the given demonstration trajectories are defined in joint space. This is motivated by the fact that most robot skills may still depend on task space variables (e.g. the manipulated objects pose), despite the same skill is also required to imitate particular robot postures.

As we are interested in whole-body obstacle avoidance, we can leverage the forward kinematics model to access the Cartesian position of different points on the robot (e.g., its joints locations). Therefore, we use a set of $M$ forward kinematic functions $f_{FK}(\eta_{\alpha}^1, \eta_{\alpha}^2, \ldots, \eta_{\alpha}^\eta)$, where $\eta_{\alpha}^1, \eta_{\alpha}^2, \ldots, \eta_{\alpha}^\eta$ elements of the joint-values vector $\eta_{\alpha}(z)$, and $M$ is the number of considered points on the robot. Note that for certain points on the robot structure, the forward kinematics only needs a subset of the joint values. For simplicity, we consider $M$ to be equal to the number of robot joints plus the end-effector (i.e. $M = \eta + 1$). Then, the full forward kinematic function $f_{FK}$ is defined as

$$
f_{FK}(\eta_{\alpha}) = [f_{FK}^{1,M}(\mu_{\alpha}^1), \ldots, f_{FK}^{M}(\mu_{\alpha}^M), f_{FK}^{M}(\mu_{\alpha}^M)]^T,
$$

where given the joint value vector $\eta_{\alpha}$ as input, all the functions compute the corresponding position $p_m$ of the $m$th point on the robot, except the last function $f_{FK}^{M}$ which also provides both the position $p_{ee}$ and the orientation $q_{ee}$ of the end-effector.

Furthermore, the volume measure $\mathcal{V}$ in Eq. (27) uses the Jacobian of the full forward kinematics function, which is defined as

$$
J_{FK}(\mu_{\alpha}) = [J_{p_1}, \ldots, J_{p_M}, J_{p_{ee}}, J_{q_{ee}}]^T,
$$

where $J_{p_1}$ and $J_{q_1}$ are the linear and angular components of the corresponding Jacobians.

3.2.2 Induced Riemannian metric: With the new integrated forward kinematic layer, we can calculate a pullback metric that directly uses task space information. This adds an additional step in the formulation of the Riemannian metric, which now requires the Jacobian of the forward kinematics $J_{FK}$ as well as the Jacobians of the VAE decoder $J_{\mu_{\alpha}}$ and $J_{\sigma_{\alpha}}$, computed from the mean and variance decoder networks. Using these two Jacobians the metric can be defined as

$$
M^\Theta(z) = M^\Theta_{\mu_{\alpha}}(z) + M^\Theta_{\sigma_{\alpha}}(z)
$$

(28)

with,

$$
M^\Theta_{\mu_{\alpha}}(z) = (J_{FK}(\mu_{\alpha}(z))^T J_{FK}(\mu_{\alpha}(z)))^{-1} J_{FK}(\mu_{\alpha}(z))^T (J_{FK}(\mu_{\alpha}(z))^T J_{FK}(\mu_{\alpha}(z)))^{-1} (J_{FK}(\mu_{\alpha}(z))^T J_{FK}(\mu_{\alpha}(z)))
$$

$$
M^\Theta_{\sigma_{\alpha}}(z) = (J_{FK}(\mu_{\alpha}(z))^T J_{FK}(\mu_{\alpha}(z)))^{-1} J_{FK}(\mu_{\alpha}(z))^T (J_{FK}(\mu_{\alpha}(z))^T J_{FK}(\mu_{\alpha}(z)))
$$

Similarly to our Riemannian metric in task space, this new metric $M^\Theta(z)$ takes large values in regions with little or no data, so that geodesics avoid passing through them. Therefore, geodesic curves generated via Eq. (28) allow us to reproduce joint space robot skills.

*We did not model the robot joint space as a high-dimensional torus for simplicity. However, we showed that our approach can easily encode data on Riemannian manifolds as in the task space case.*
4 Geodesic motion skills

As explained previously, geodesics follow the trend of the data, and they are here exploited to reconstruct motion skills that resemble the human demonstrations. In this section, we describe geodesic computation in both settings, namely, where the VAE is trained on task space or joint space trajectories. Moreover, we explain how new geodesic paths, that avoid obstacles on the fly, can be obtained by metric reshaping. 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.

4.1 Generating motion:

Geodesic curves generally follow the trend of the demonstrations data, due to the role of uncertainty in the metric. Specifically, the Riemannian metrics Eq. (10) and Eq. (28) tell 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 (see Fig. 1). This behavior is due to the way that the metric \( M \) is defined: Our Riemannian metric \( M \) 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 \( M \), which is a function of \( M \), 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 a Euclidean metric (i.e., an identity matrix), geodesics correspond to straight lines. Such geodesics certainly neglect the data manifold geometry.

Noted that geodesics do not typically follow a closed-form equation on these learned manifolds, and numerical approximations are required. This can be done by direct minimization of curve length (Shao et al. 2018; Kalatzis et al. 2020), A* search (Chen et al. 2019), integration of the associated ODE (Arvanitidis et al. 2019), or various heuristics (Chen et al. 2018). In this paper, we compute geodesics on \( M \) by approximating them by cubic splines \( c \approx \omega_\lambda(z_c) \), with \( z_c = \{z_{c0}, \ldots, z_{ck}\} \), where \( z_{ck} \in \mathcal{Z} \) is a vector defining a control point of the spline over the latent space \( \mathcal{Z} \). Given \( K \) control points, \( K - 1 \) cubic polynomials \( \omega_{\lambda_i} \), with coefficients \( \lambda_{0,0}, \lambda_{1,1}, \lambda_{1,2}, \lambda_{1,3} \) have to be estimated to minimize its Riemannian length,

\[
\mathcal{L}_{\omega_\lambda}(z_c) = \int_0^1 \sqrt{\langle \omega_\lambda(z_c), M(\omega_\lambda(z_c))\omega_\lambda(z_c) \rangle} dt.
\]  

The resulting geodesic \( c \) computed in \( \mathcal{Z} \) is used to generate the robot motion by decoding it through the VAE networks \( \mu_{\psi} \) and \( \mu_{\psi} \) or \( \mu_{\phi} \) depending on the ambient space setting. The obtained trajectory is then executed on the robot arm to reproduce the required skill. In the task space setting, the decoded geodesics can be deployed on the robot using a Cartesian impedance controller or inverse kinematics. In the joint space setting, the decoded geodesics can be employed directly on the robot as a joint trajectory reference to be tracked by joint position or impedance controllers.

4.2 Geodesics in task space \( \mathbb{R}^3 \times S^3 \):

In this section, we investigate the geodesic motion generation in task space. To illustrate the motion generation mechanism, we consider a simple experiment where the demonstration data at each time point is confined to \( \mathbb{R}^2 \times S^2 \), i.e. only two-dimensional positions and orientations are considered. We artificially create position data that follows a J shape and orientation data that follows a C shape projected on the sphere (see Fig. 4). We fit our VAE model to this dataset, and visualize the corresponding latent space in Fig. 3, where 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. 3 shows the same embedding but with a background color proportional to \( \log \sqrt{\det M} \). This quantity, known as the magnification factor (Bishop et al. 1997), generally takes large values in regions where distances are large, implying that geodesics will try to avoid such regions. In Fig. 3, 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 hence resemble the training data patterns. In contrast, Euclidean geodesics (red curves in the figure) fail to stay in the data manifold. Our proposal is to use these geodesics on the learned manifolds as our robot motion generation mechanism.

Note that both panels in Fig. 3 depict two distinct horseshoe-like clusters, which is a result of the antipodality of the data in \( S^2 \). More precisely, the bimodal distribution of the antipodal quaternion data is encapsulated by these two clusters in the latent space. In practical settings, as long as the geodesic curve does not cross across clusters (both start and goal points belong to the same cluster), the quaternion sign is unchanged. We experimentally examine this in Section 5.

4.3 Geodesics in joint space \( \mathbb{R}^9 \):

In this section, we investigate the geodesic computation for joint space movements. We use a toy example where a 2-DOF robot arm follows an S-shaped trajectory in task space using two different joint configurations (i.e., two different inverse-kinematics solutions), as shown in Fig. 5—left. We observe two sets of demonstrations in joint space that reproduce the same end-effector movements when applied to the robot. We can also see in the middle and right panels of Fig. 5 the geodesics computed using the Riemannian and Euclidean metrics, depicted as blue and red curves, respectively. The background of the middle panel illustrates the predictive uncertainty over the latent space \( \mathcal{Z} \), where we again see low uncertainty near the data, and high otherwise. For completeness, the background in the right panel illustrates the magnification factor. The latent mean embedding of the training data are depicted as semi-transparent white points. Similar to the previous section, the geodesics generated using the Riemannian metric stay within the ‘boundary’ near the training data.

Furthermore, it is easy to note that the learned manifold comprises two clusters, but unlike the previous task space example, these clusters arise from the two different joint space solutions provided in the training data. This indicates that the clusters in the learned manifold encapsulate the provided solutions in the demonstrations. When the number of clusters grows, the geodesic has a higher chance to travel among them to find a path with minimal energy as the high-energy regions may become narrow. However, unnecessary frequent switching among these clusters may often lead to jerky geodesics, therefore negatively impacting the geodesics quality, particularly in robots with a high degree of freedom (e.g. DOF \( \geq 7 \)). Later in Section 5, we experimentally show that this issue can be alleviated by increasing the latent space dimensionality.

4.4 Obstacle avoidance using ambient space metrics

Often human demonstrations do not include any notion of obstacles in the environment. Therefore, obstacle avoidance is usually treated as a separate 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 is that the robot is still unable to avoid unseen obstacles on the fly.

Our Riemannian approach provides a natural solution to this problem. The learned metrics in latent space Eq. (10) and Eq. (28) measure the length of a geodesic curve under the Euclidean space of the ambient space \( \mathcal{X} \). We can easily modify this to account for unseen and dynamic obstacles. Intuitively, we can increase the length of curves that intersect obstacles, such that geodesics are less likely to go near them. Next, we explain how obstacle avoidance can be achieved for both task space and joint space settings.

4.4.1 Obstacle avoidance in task space \( \mathbb{R}^3 \times S^2 \): Here we explain how we can reshape the learned metric to avoid obstacles in the task space setting, where only the robot end-effector is considered. Formally, we propose to define the ambient metric of the end-effector position to be

\[
M^a_\alpha(x) = \left(1 + \zeta \exp \left( -\frac{||x - o||^2}{2r^2} \right) \right) \mathbb{I}_3, \quad x \in \mathbb{R}^3,
\]

where \( \zeta > 0 \) scales the cost, \( o \in \mathbb{R}^3 \) and \( r > 0 \) represent the position and radius of the obstacle, respectively. For the orientation component, we assume a flat ambient metric \( M^a_\alpha(x) = I_3 \). Under this new ambient metric, geodesics will generally avoid the obstacle, though we emphasize this is only a soft constraint. This approach is similar in spirit to CHOMP (Ratliff et al. 2009) except our formulation works along a low-dimensional learned manifold, whose solution is then decoded to the task space of the robot.

Under this ambient metric, the associated (reshaped) Riemannian metric of the latent space \( \mathcal{Z} \) becomes,

\[
M(z) = M^a_\alpha(z) + M^a_\sigma(z) + M^a_\mu(z) + M^a_\kappa(z),
\]
with 
\[ M^\sigma(z) = J_{\mu}(z)^T M^\sigma_{\alpha}(\mu(\theta)) J_{\mu}(z), \]
\[ M^\kappa(z) = J_{\sigma}(z)^T M^\kappa_{\alpha}(\mu(\theta)) J_{\sigma}(z), \]
\[ M^\eta(z) = J_{\mu}(z)^T M^\eta_{\alpha}(\mu(\theta)) J_{\mu}(z), \]
\[ M^\phi(z) = J_{\alpha}(z)^T M^\phi_{\alpha}(\mu(\theta)) J_{\alpha}(z), \]
where \( M^\sigma \) and \( M^\kappa \) represent the position and orientation components of the obstacle-avoidance metric \( M_X \), respectively. 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. As stated before, obstacles can be avoided only by the end-effector under this task space setting. Next we explain how we can extend this approach so that the robot can move away from obstacles using its whole body.

4.4.2 Obstacle avoidance in joint space \( \mathbb{R}^n \): Avoiding obstacles at the robot link level while performing motion skills requires to consider the whole robot kinematic structure. Classical motion planning methods model the obstacles geometry into the configuration space, and later compute an obstacle-free path via sampling methods (Elbanhawi and Simic 2014). In contrast, we take advantage of the forward kinematics layer (see Fig. 7-bottom), which provides us with task space poses of any point on the robot body, to compute an obstacle-avoidance ambient metric. Similar to the task space formulation presented previously, this ambient metric is then exploited to reshape the learned metric and generate modified geodesic curves that produce collision-free robot movements.

Specifically, we need to define a collection of points on the robot body \( p_1, \ldots, p_M \) with \( p_m \in \mathbb{R}^3 \). These points are then used to compute the ambient space metric for obstacle-avoidance purposes. Therefore, a larger collection of points provides a more robust obstacle-avoidance performance at the cost of higher computational complexity. Given the set of points of interest, we compute an associated ambient metric following Eq. (30) with \( x = p_m \). Similar to the task space setting, since the orientation of obstacles is not considered, the corresponding ambient space metric is an identity matrix. Finally, we form the whole ambient metric as \( M_X = \text{blockdiag}(\{ M^p_{X1}, M^p_{X2}, \ldots, M^p_{XM} \}) \), which is then used to reshape the learned metric of Eq. (28) as,
\[ M(z) = M^\theta(z) + M^\phi(z), \]
with,
\[ M^\theta(z) = (J_{f_{\alpha}}(\mu(\theta)) J_{\alpha}(z))^T M_X (J_{f_{\alpha}}(\mu(\theta)) J_{\alpha}(z)), \]
\[ M^\phi(z) = (J_{f_{\alpha}}(\mu(\theta)) J_{\alpha}(z))^T M_X (J_{f_{\alpha}}(\mu(\theta)) J_{\alpha}(z)). \]

4.4.3 Generating geodesics on discrete manifolds: Having proposed the robot motion generation as the computation of geodesic curves, we evidently need a fast and robust algorithm for computing geodesics to make our method practical. As we work with low-dimensional latent spaces, we here propose to simply discretize them on a regular grid and use a graph-based algorithm for computing shortest paths. Specifically, we create a uniform grid over the latent space \( \mathcal{Z} \), and assign a weight to each edge in the graph corresponding to the Riemannian distance between neighboring nodes (see Fig. 6). Geodesics are then found using Dijkstra’s algorithm (Cormen et al. 2009). This algorithm selects a set of graph nodes,
\[ G_c = \{ g_0, g_1, \ldots, g_{s-1}, g_s \}, \quad g_s \in \mathbb{R}^D, \]
where \( g_0 \) and \( g_S \) 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, computed as in Eq. (6). To ensure a smooth trajectory, we fit a cubic spline \( \omega_3 \) to the resulting set \( G_c \) by minimising the mean square error. The spline computed in \( Z \) is finally used to generate the robot motion through the mean decoder networks: \( \mu_3 \) and \( \mu_\phi \) or \( \mu_\theta \). 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. 6). This way we avoid querying 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.

## 5 Experiments

In this section, we showcase the capabilities of our method in different experiments where human demonstrations are provided in both task and joint spaces. We provide a full description of the experimental setup, which includes the design of the VAE networks, Riemannian manifold learning, real-time geodesic computation, and calculation of the ambient metric to avoid obstacles. We evaluate the performance of our approach using real-robot experiments, namely, Reach-to-grasp and Pouring tasks. The Pouring tasks is primarily designed to demonstrate the model capability to generate multiple-solution trajectories and avoid obstacles under the task space setting. The Reach-to-grasp task is intended to demonstrate the aforementioned features under a joint space setting. Furthermore, the video is available at: https://sites.google.com/view/motion-generation-on-manifolds/home.

### 5.1 Setup description

We consider a set of demonstrations involving a 7-DOF Franka Emika Panda robot arm endowed with a two-finger gripper. The demonstrations were recorded using kinesthetic teaching at a frequency of 10Hz. We calculate geodesics on \( 100 \times 100 \) and \( 50 \times 50 \times 50 \) graphs under task space and joint space settings, respectively. Our Python implementation runs at 100Hz on ordinary PC hardware. The approach readily runs in real time. Additionally, obstacles are simulated along with a digital twin of the robot in a simulated environment to provide real-time obstacle information (we did not integrate obstacle localization systems in our setups).

### 5.2 Architecture

In this section, we describe the VAE architecture under both \( \mathbb{R}^3 \times S^3 \) and \( \mathbb{R}^3 \) settings. The VAE architectures are implemented using PyTorch (Paszke et al. 2019).

#### 5.2.1 VAE architecture in \( \mathbb{R}^3 \times S^3 \): This particular VAE network is design to reconstruct end-effector poses in \( \mathbb{R}^3 \times S^3 \). The overall architecture is depicted in Figure 7-top with the different components that are required to correctly reproduce end-effector poses. In this architecture, both the decoder and encoder networks have two hidden layers with 200 and 100 neuron units (depicted as beige boxes) which output the mean and variance vectors (depicted as orange boxes). Furthermore, as previously explained, the variance and concentration parameters for position and quaternion data are estimated using RBF decoder networks with 500 kernels calculated by \( k \)-means over the training dataset (Arvanitidis et al. 2018) with predefined bandwidth. In this particular setup, the VAE uses a 2-dimensional latent space (depicted by the green box) to encode the 7-dimensional input vector (depicted as blue boxes on the left). Moreover, the VAE reconstructs the encoded input back to the ambient space (depicted as blue boxes on the right) using the decoder networks.

In this setting, the VAE is implemented using a single neural-network as the decoder mean for the position and orientation, while the variance and concentration networks are implemented separately. As a result, the learned metric is defined as,

\[
M(z) = M^x_\mu(z) + M^x_\sigma(z) + M^\theta_\mu(z), \tag{33}
\]

with

\[
M^x_\mu(z) = J_{\mu_\phi}(z)^T M_\Omega(\mu_\phi(z)) J_{\mu_\phi}(z),
M^x_\sigma(z) = \begin{bmatrix} M_{\phi}(\mu_\phi(z)) & 0 \\ 0 & M_\Omega(\mu_\phi(z)) \end{bmatrix},
M^\theta_\mu(z) = J_{\theta_\phi}(z)^T M_{\theta}(\mu_\phi(z)) J_{\theta_\phi}(z),
M^\theta_\phi(z) = J_{\theta_\phi}(z)^T M_\Omega(\mu_\phi(z)) J_{\theta_\phi}(z),
\]

where \( J_{\mu_\phi} \in \mathbb{R}^{(D_x+D_\theta) \times d} \) is the Jacobian of the joint decoder mean network (position and quaternion), and \( J_{\sigma_\theta} \in \mathbb{R}^{D_x \times d} \) and \( J_{\kappa_\phi} \in \mathbb{R}^{D_\theta \times d} \) are the Jacobians of
Figure 8. Left: The yellow curves in the right cluster show geodesics starting from the same points and ending up at random targets, and the blue curves in the left cluster connect random points on the manifold. The background depicts the magnification factor derived from the learned manifold, and the semi-transparent white points show the encoded training dataset. Right: The decoded geodesic employed on the robot.

Figure 9. The evolution of the individual quaternion components of the green geodesic as it passes through high-energy regions.

The International Journal of Robotics Research XX(X)

the decoder variance and concentration RBF networks, respectively. 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 $S^3$ to then define the corresponding von Mises-Fischer distribution as in Eq. (15). The yellow arrow in Figure 7-top shows the flow of the data from ambient space back to the latent space in order to compute the pullback metric.

The ELBO parameters $\beta_1$ and $\beta_2$ in Eq. (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. The same architecture is used for all the experiments.

5.2.2 VAE architecture in $\mathbb{R}^7$: Here, we describe our VAE network that reconstructs joint space movements in $\mathbb{R}^7$. The overall architecture is depicted in Figure 7-bottom with different components. The input vector (depicted as blue box on the left) is a joint-value vector representing a single configuration of the robot arm on a trajectory. This vector is fed to the encoder network with two hidden layers of 200 and 100 neuron units (depicted as beige boxes) which are the mean and variance vectors for the latent variable (depicted as orange boxes). Moreover, the variance RBF decoder network uses 500 kernels calculated by $k$-means over the training with predefined bandwidth.

Under the joint space setting, the VAE uses a 3-dimensional latent space $\mathcal{Z}$ to encode the input vectors $\theta$. As usual, the decoder network reconstructs the encoded inputs back to the joint space. However, in order to access the task space information, necessary for whole-body obstacle avoidance, our architecture is integrated with a forward kinematics layer (depicted as gray box). Note that this layer is predefined based on the robot arm kinematic model and does not change during training. We leverage this layer to compute task space information regarding multiple points on the robot (and not just the end-effector) given the input configuration vector $\theta$. To implement this component we used the Python Kinematic and Dynamic Library PyKDL (Orocos 2021).

It is worth noting that as this VAE architecture uses the forward kinematics layer during training, singular kinematic configurations need special attention. The main problem arises in the formulation of our ELBO in Eq. (26), which uses the volume measure computed as a function of the determinant of the Jacobian of the forward kinematics function. We can detect singularities when $\det(J_{\text{FK}}(\alpha_{\theta}(z_i))) = 0$, which may occur due to random initialization of the VAE. To circumvent this issue and guarantee that the learning process is not disrupted, we simply add a small regularization term to the kinematic Jacobian.

Finally, we evaluate the obstacle avoidance capabilities in different scenarios where the obstacles partially or entirely obstruct the solutions in joint space. In these experiments, the ambient metric of Eq.(32) is formulated by considering all the joints on the robot, in addition to the end-effector. This ensures that the robot will avoid the obstacles as long as they obstruct the solution for one or more joints or the end-effector. In other words, we do not consider points on the robot links lying between joints, since it was not necessary in our experiments. However, extra points on the robot links can be easily added to guarantee a more robust obstacle avoidance using the whole robot body.

5.3 Experiments in $\mathbb{R}^3 \times S^3$

The first set of the experiments focuses on tasks where only the robot end-effector motion is relevant for the task, therefore the demonstrations are recorded in $\mathbb{R}^3 \times S^3$.

5.3.1 Reach-to-grasp: The first set of experiments is based on a dataset collected while an operator performs kinesthetic demonstrations of a grasping skill. The grasping motion includes a 90° rotation when approaching the object for performing a side grasp (see Fig. 8). The demonstration trajectories start from same end-effector pose, and they reach the same target position in task space. To reproduce this grasping skill, we computed a geodesic in $\mathcal{Z}$ which is decoded to generate a continuous trajectory in $\mathcal{X}$, which
closely reproduces the rotation pattern observed during demonstrations. Figure 8–left 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 and blue curves are geodesics between points assigned from the start and endpoints of the demonstrations. The top panel in Fig. 8 shows geodesics in two different scenarios: The yellow geodesics in the right cluster start from same pose and end up at different targets, while the blue geodesics in the left cluster start and end in different random targets. The results show that the method can successfully generate geodesics that respect the geometry of the manifold learned from demonstration.

As expected, the magnification factor (Fig. 8–left) shows that the learned manifold is composed of two similar clusters, similarly to the illustrative example in Fig. 3. We observed that this behavior emerged due to the antipodal encoding of the quaternions, where each cluster represents one side of the hyper-sphere. To confirm this, we generated a new geodesic, depicted in green in the top panel of Fig. 8, which is designed to cross the clusters boundaries (start and end locations belong to different clusters). Figure 9 depicts the evolution of the quaternion components corresponding to decoded green geodesic. As this geodesic curve crosses the clusters, the sign of the end-effector quaternion flips (highlighted by the black rectangle). It is worth emphasizing that by staying on the manifold and avoiding these boundaries, no post-processing of raw quaternion data is required during training or reconstruction. This can be easily guaranteed by monitoring the energy along the curves, which indicates when geodesics approach a high-energy region. For instance, the average energy of the blue and yellow geodesics are 7.50 and 10.51, respectively, while that of the green geodesic is $2.49 \times 10^3$. As a result, we can simply identify and avoid these geodesics.

Figure 8–right shows the reconstructed geodesic executed by the robot, where the overlapping images display the time evolution of the skill. It is easy to observe that the desired motion is successfully generated by our model. Note how the end-effector orientation evolves on the decoded geodesic in the ambient space, showing that the $90^\circ$ rotation is properly encoded and reproduced using our approach.

5.3.2 Pouring: To evaluate our model on a more complicated scenario, we collected a dataset of pouring task demonstrations. The task involves grasping 3 bottles from 3 different positions and then pouring at placed at 3 different locations. The demonstrated trajectories cross each other, therefore providing a multiple-solution setting. As a result, with 3 sets of demonstrations, all 9 permutations for grasping any bottle from the table and then pouring at any cup are feasible, despite only a small subset of them are demonstrated.

The first feature we want to test in this setting is the obstacle avoidance capabilities via metric reshaping. To do so, we compute the ambient space metric in Eq. (30) based on a spherical obstacle that partially blocks the low-energy regions that the geodesics exploit to find a solution. This way the geodesics are forced to either use the low-energy regions that the individual demonstrations provide or improvise and find a hybrid novel path based on a subset of demonstrations. Figure 10–left shows the geodesic performing obstacle avoidance around a moving obstacle while following the geometry of the manifold. Two time instances of the obstacle are depicted as red and yellow circles in the latent space for illustration purposes. The red and yellow curves represent geodesics avoiding the red and yellow obstacles, respectively. These curves correspond to one time frame of the adapted geodesic, showing how our method can deal with dynamic obstacles. Fig. 10–right shows the decoded geodesics performed on the robot, where transparent robot arms show the temporal evolution of the skill. In order to correctly perform obstacle avoidance, the parameter $x$ in ambient space metric in Eq. (30) represents the position of the bottle when grasped and the obstacle radius $r$ is modified to account for the bottle. To do so, we simplify the bottle geometry by approximating it using a sphere and add its radius $r_{\text{bot}}$ to the radius of the obstacle as $r = r_{\text{obs}} + r_{\text{bot}}$, where $r_{\text{obs}}$ is the obstacle radius. This prevents the bottle from colliding with the yellow and red spheres that represent the obstacle.

Figure 11–left shows the ability to leverage multiple-solution tasks to generate novel movements emerging as combinations of the observed demonstrations. Specifically, we generate a combination of three geodesics that can be used to pour all the cups with a single bottle. The geodesic (depicted as yellow curve) starts from an initial point (depicted in green) in the white-demonstrations group at the top, grasps the bottle, pours the first cup using the green demonstration group, then goes to the second cup using the blue-demonstrations group, and finally pours the third cup.
Figure 11. Left: The geodesic shown as the yellow curve combines the blue, green, and white demonstration groups to form a hybrid solution. This experiment uses three geodesics to pour all the cups using a single bottle. Right: The decoded geodesics performed on the robot depicted by superimposing images from different time frames. The transparent robot arms depict the trace of the motion trajectory, which begins with pouring the right cup, the middle, and lastly the left cup.

Figure 12. Left: The geodesics calculated in 2-dimensional latent space, depicted as the yellow curve, reveal several unnecessary transitions between different solutions when connecting two points in the same demonstration. Right: Geodesics computed in 3-dimensional latent space, shown by the yellow curve, shows that the geodesic does not switch between clusters.

Figure 13. Concept drawing. Left: The hollow tube represents the metric, with low-energy regions inside surrounded with high-energy boundaries. The geodesic depicted as blue curve travels through the low-energy regions. Right: The hollow tube is partially blocked by a solid high-energy region corresponding to the ambient space metric representing the obstacle. The geodesic depicted as blue curve successfully travels through the low-energy regions meanwhile avoiding the obstacle.

5.4 Experiments in the joint space $\mathbb{R}^\eta$

In this section, we focus on tasks where joint-level motion patterns are relevant, and therefore the human teacher provides kinesthetic demonstrations in $\mathbb{R}^\eta$, where $\eta$ is the number of DOF of the robot. When learning Riemannian metrics in this setting, we initially designed a 2-dimensional latent space for our VAE, which proved to be insufficient to encode the demonstrated motion patterns. Specifically, we analyze the capacity of the latent space to encode the skill manifold experimentally. To do so, we investigated the switching behavior of geodesics in $\mathbb{Z}$. In other words, overlap between low-energy regions in $\mathbb{Z}$, representing two or more different demonstration sets, may lead to unnecessary and frequent switches between these solutions when computing a geodesic. To put it differently, when computing a geodesic, switching between two demonstration sets is unavoidable when the total energy of the geodesic switching between them is less than the total energy without the switch. However, switching behaviors may be avoided by having high-energy regions among demonstration clusters in $\mathbb{Z}$. Furthermore, specifically under the joint space setting, the frequent switching in geodesics may lead to jittery motion in task space when applied to the robot. To illustrate this issue, we designed a simple experiment in which a robotic arm follows a circular pattern with its end-effector. The start and target configuration of the geodesic are selected from the same demonstrated trajectory, and the objective is to evaluate if the geodesic stays on the low-energy regions corresponding to the same demonstrated trajectory.

Fig. 12–left shows a geodesic curve computed in the 2-dimensional latent space depicted as the yellow curve. This geodesic exhibits unnecessary switches among different solutions (i.e. circular white-demonstrations). Therefore, when the decoded geodesic is deployed on the robot, it results in jerky movements and undesirable back-and-forth motions. To solve this issue, we evaluated the same
Figure 14. **left**: The magnification factor in 3-dimensional latent space contains hollow tubes representing the learned Riemannian metric. These hollow tubes contain low energy regions surrounded by high energy boundaries. Several geodesics are calculated and visualized in blue. The manifold’s low energy areas are rendered transparent. **middle**: The same magnification factor when an obstacle is introduced in such a manner that all viable solutions in the ambient space are blocked. Due to the fact that the obstacle introduces a high energy zone (in red) that goes through all of these hollow tubes, none of the geodesics are feasible, therefore no geodesic is shown in this plot. **right**: The same magnification factor when an obstacle is introduced in such a manner that partially obstruct the solutions in the ambient space. Several geodesics (blue curves) were left outside of the obstacle region on the left side of the panel.

**Figure 15.** **Left**: The decoded geodesic employed on the robot arm when no obstacle is present in the environment. **Right**: The decoded geodesic employed on the robot arm with an obstacle partially obstructing the solutions. The obstacle is depicted as the green sphere.

experiment using a 3-dimensional latent space. Figure 12–right shows the magnification factor of the metric learned using the same training data but in a 3-dimensional latent space. This magnification factor shows several torus-like clusters, representing separate demonstration groups instead of collapsing them into a plane, as in the 2-dimensional case. Moreover, the resulting geodesic (depicted as the yellow curve) does not switch among solutions, which provides stable and smooth robot end-effector movements when decoded.

To provide further details on the learned manifold in the 3-dimensional latent space, we create an illustration shown in Fig. 13, where the learned metric is shown as yellow hollow tubes. Their inner part encodes low-energy regions which are surrounded by high-energy boundaries. Figure 13 displays an horizontal cut to show the inner part of the learned metric. To illustrate the obstacles, the right hollow tube is partially blocked by a red sphere representing the ambient space metric in the latent space $\mathcal{Z}$, which is a solid high-energy region. Additionally, the figure depicts a geodesic curve traveling successfully along both tubes, showcasing a collision-free trajectory at the right-side plot.

**5.4.1 Reach-to-grasp:** Similar to the task space setting, we used the reach-to-grasp task to evaluate our approach under the joint space setting. In this case the demonstrations are quite similar at the end-effector level but differ at the joint space, as we exploited the kinematic robot redundancy to provide different joint trajectories. Figure 14–left shows the magnification factor in a 3-dimensional latent space, where it can be seen that the learned metric corresponds to several connected and separated hollow tubes. As mentioned previously, these hollow tubes have low-energy inner regions surrounded by high-energy boundaries, analogous to the learned metric in a 2-dimensional latent space. As shown in the figure, the generated geodesics stay inside the tubes and avoid crossing the boundaries. Figure 15–left shows the robot executing the decoded geodesic by applying the joint values directly on the robot using a joint position controller. We can observe that the decoded geodesic is able to generate the demonstrated grasp motion with $90^\circ$ rotation during the approaching part. Note that the start and end points of the geodesics are extracted from the demonstrations.

Figure 14–middle displays the magnification factor where an obstacle is placed in such a way that all possible solutions in joint space are blocked (e.g. obstacle placed on the common target of all the demonstrations). Note that the obstacle introduces a high energy zone (depicted in red) that passes through all of the hollow tubes representing the learned manifold, therefore, none of the geodesics are
feasible. Additionally, Fig. 14–right illustrates the same learned metric but reshaped using a different ambient space metric. We can now see that the obstacle partially obstructs the possible solutions, and therefore some few geodesics (depicted as blue curves) are still able to successfully generate obstacle-free movements. Figure 15–right shows the robot executing the decoded geodesic using a joint position controller while avoiding the obstacle.

We designed another experiment to showcase how the multiple-solution capabilities can be leveraged to generate collision-free movements. If the different demonstrated joint space trajectories sufficiently overlap, the learned manifold will be characterized by several overlapping low-energy regions (i.e., hollow tubes), which geodesic curves can travel through. As a result, if the obstacles partially block the learned manifold, the geodesic may still travel among solutions to generate new movements out of combinations of the provided demonstrations. To show this behavior, we used a different demonstrations dataset where we accounted for several overlapping solutions starting from the same joint configuration. Figure 16–left shows the magnification factor of the learned Riemannian metric in the 3-dimensional latent space. The learned manifold and the ambient metric can be distinguished visually based on their energy values. The ambient metric, which represents the obstacle regions (depicted in red), encodes high-energy zones. The learned manifold (depicted in yellow) is characterized by several entangled hollow tubes. The geodesic, shown as the blue curve, begins from the common start configuration on the left and successfully navigates around the obstacle to reach the target on the right side. Figure 16–right displays the decoded geodesic deployed directly on the robot arm using a joint position controller. Note how the robot arm avoids the obstacle by carefully maneuvering around it while successfully performing the grasp skill.

5.4.2 Pouring: To evaluate the method in a more complex scenario, we also performed the pouring task under the joint space setting. Similar to the task space experiment, the demonstrations also overlap in joint space. For this specific experiment, we mainly focus on evaluating the multiple-solution trajectories. Figure 17–left shows the magnification factor in 3-dimensional latent space showing entangled hollow tubes, which represent the learned Riemannian metric. Each hollow tube encodes low-energy regions surrounded by high-energy boundaries, the latter outlined by green, red, and blue solid lines. Each color represents one group of demonstrations. Since the magnification factor learned with the original dataset of 5 demonstrations per group (corresponding to the bottle initial position) is very hard to visualize, we opt for simplicity and used a subset of 2 demonstrations per bottle. As shown, the geodesic (depicted as the black curve) starts from a point in the second demonstration group (green border) and switches to the first group (red border) to reach the target located in the third group (blue boundary). Figure 17–right shows that the decoded geodesic employed on the robot successfully performs the pouring task. It is also evident the geodesic uses a multiple-solution strategy to reproduce a new trajectory.
that was not explicitly demonstrated to the robot in the training phase.

6 Discussion

We presented a novel LfD approach that learns Riemannian manifolds from human demonstrations, over which geodesic curves are computed and used as a motion generator. These geodesics are capable of recovering and synthesizing robot movements similar to the provided demonstrations. Therefore, our technique enables robot motion planning between two arbitrary points on the learned manifold via geodesics. This is achieved through a variational autoencoder (VAE) from which a pullback Riemannian metric can be computed in the VAE latent space and used to generate movements in both task and joint spaces. Additionally, we proposed reshaping the learned Riemannian metric using ambient metrics to avoid static or dynamic obstacles on the fly. This metric reshaping along with a forward kinematics layer enabled multiple-limb obstacle avoidance capabilities, which allowed us to achieve reactive motions. In order to guarantee fast geodesic computation and motion generation, we developed a graph-based technique built on the Dijkstra algorithm to provide real-time motion synthesis and adaptation. We extensively tested our approach in both joint and task space settings, showing that our geodesic motion generation performs effectively in both settings and in a variety of tasks such as grasping and pouring.

Geodesics generalization: During our experiments, we observed that when the geodesic curves are forced to leave the learned manifold, thus crossing high-energy boundaries, the parts of the geodesic that lie outside the manifold may cause inaccurate and undesirables motions. This is a potential issue when the given start or target points are placed outside the data manifold, or when the obstacles fully blocked it, which forces the geodesics to leave the manifold to comply with the desired task specifications. We believe this is a consequence of using VAEs to learn the skill manifold where the data lying outside the training data support may be arbitrarily misrepresented in the latent space $\mathcal{Z}$. We hypothesize that this problem may be addressed by 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.

Obstacle avoidance: Regarding our obstacle avoidance approach, the ambient metric used to reshape the learned metric enforces a “soft constraint” rather than a “hard constraint”. This means, under certain situations, the geodesic might still cross the obstacle instead of avoiding it. It is worth noting that we were unable to replicate this scenario since the generated geodesics tended to avoid the obstacle by abandoning the manifold rather than crossing the obstacle, which can be easily detected and prevented if necessary. To address this potential issue, the graph-based geodesic can be further exploited by removing the nodes near the obstacle from the graph instead of re-weighting the edges. While this strategy may reduce the computational overhead and work in practice, it is not theoretically grounded. Finally, our obstacle avoidance formulation only considered simple obstacles, but the strategy can be extended to multiple dynamic obstacles. Instead of working with single 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 reasonable.

Latent space topology: As explained in Section 5, we observed that the latent space dimensionality plays a critical role when learning joint space motions. As reported, a 3-dimensional latent space was necessary to calculate smooth geodesic curves for joint space tasks. This resulted in increased computational complexity when calculating the ambient metric and geodesics in the latent space. This implies that this issue may exacerbate when working with high-DOF robots, requiring the use of higher-dimensional latent spaces, and therefore raising the need to design more efficient geodesic computation methods. Note that this phenomenon may be related to the latent space topology, which is here assumed to be Euclidean in accordance to the used VAE Gaussian prior. We plan to theoretically and experimentally analyze the effects of the latent space topology when learning Riemannian manifolds for robot motion generation.

References


