Riemannian Manifold Learning for Robot Motion Skills

Hadi Beik-Mohammadi[1], Leonel Rozo[1], Gerhard Neumann[2], Søren Hauberg[3]

I. INTRODUCTION

Bringing geometric methods to robot learning provides an alternative view of how data may be processed, encoded, and synthesized. This opens the door to investigate and exploit the interplay of geometry, machine learning, and control for learning complex robot skills. When learning robot motion skills, one can assume that observed (high-dimensional) data may lie on a lower intrinsic dimensionality due to constraints on the way the data is generated. For example, a robot arm is constrained in its set of feasible configurations, which may imply that end-effector trajectories lie on a low-dimensional manifold. To build effective geometry-aware motion skills, we can learn a skill manifold from demonstrations, which are encoded in a low-dimensional latent space. Subsequently, we can exploit this manifold to generate new motions via geodesics while considering ambient space constraints, e.g., obstacle avoidance, on the fly.

II. MOTION SKILLS LEARNING VIA RIEMANNIAN MANIFOLDS

A. Background

1) Variational autoencoders: One probabilistic approach to learn data manifolds is *variational autoencoders (VAE)* [1], that can be seen as a probabilistic generalization of classic autoencoders [2]. The VAE realizes the generative model

$$p(\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{z}|0, I\right), \qquad \boldsymbol{z} \in \mathbb{R}^d \tag{1}$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}|\mu(\boldsymbol{z}), \sigma^2(\boldsymbol{z})\right), \qquad \boldsymbol{x} \in \mathbb{R}^D \tag{2}$$

where $d < D$. Here the latent variable $\boldsymbol{z}$ can be viewed as a lower dimensional representation of the data $\boldsymbol{x}$. In practice, the functions $\mu, \sigma : \mathbb{R}^d \to \mathbb{R}^D$ are represented by neural networks, and inference is performed variationally by maximizing the *evidence lower bound (ELBO)*

$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log(p(\boldsymbol{x}|\boldsymbol{z}))\right] - \mathrm{KL}\left(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\right), \tag{3}$$

where the approximate posterior (encoder)

$$q(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}|\mu_q(\boldsymbol{x}), \sigma_q^2(\boldsymbol{x})\right) \tag{4}$$

is also a neural network.

*This work was supported in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no 757360) and by a research grant (15334) from VILLUM FONDEN.

[1]Bosch Center for Artificial Intelligence (BCAI) (hadi.beik-mohammadi, leonel.rozo)@de.bosch.com

[2] Autonomous Learning Robots Lab, Karlsruhe Institute of Technology (KIT) gerhard.neumann@kit.edu

[3] Section for Cognitive Systems, Technical University of Denmark (DTU) sohau@dtu.dk

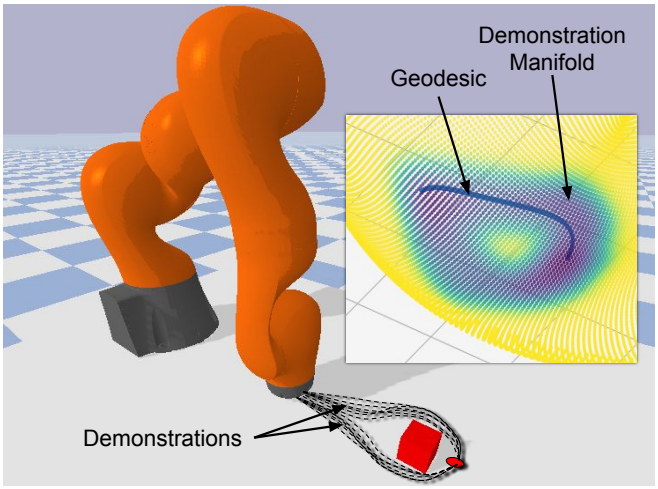

Fig. 1: Demonstration manifold generated using a VAE.

2) Riemannian geometry in VAEs: In general, the latent representation recovered by a VAE is not unique [3]. Assuming that the mean generator (decoder) network $\mu(\boldsymbol{z})$ has sufficient capacity, then the latent representation can be distorted into $\tilde{\boldsymbol{z}} = g(\boldsymbol{z})$ by an unknown function $g : \mathbb{R}^d \to \mathbb{R}^d$ and the decoder may adapt to $\mu(g^{-1}(\tilde{\boldsymbol{z}}))$. This implies that latent space distances are largely meaningless [3], which limits the value of the representation for robot learning. Arvanitidis *et al.* [4] proposed to endow the latent representation with a Riemannian metric to avoid this issue. In particular, they suggest using the pullback metric, such that distances in latent space now reflect curve lengths measured along the estimated manifold in data space. Letting f denote the mapping from representation to observation, the length of a curve $\boldsymbol{c} : [0, 1] \to \mathbb{R}^d$ in latent space can be defined as

$$\text{Length}[\boldsymbol{c}] = \int_0^1 \|\partial_t f(\boldsymbol{c}_t)\| \mathrm{d}t \tag{5}$$

and the distance between two latent points can be defined as the length of the shortest connecting path.

The generative process of a VAE can be written as

$$\boldsymbol{x} = \mu(\boldsymbol{z}) + \sigma(\boldsymbol{z}) \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I) \tag{6}$$

$$= \begin{pmatrix} I, & \mathrm{diag}(\epsilon) \end{pmatrix} \begin{pmatrix} \mu(\boldsymbol{z}) \\ \sigma(\boldsymbol{z}) \end{pmatrix} = P\ f(\boldsymbol{z}), \tag{7}$$

such that we may view the VAE as spanning a manifold (via f) in $\mathbb{R}^{2D}$, which is then randomly projected to data space (via P) [5]. Following existing literature [3], [4] we exploit f to define distances (5). If σ takes large values in regions of latent space where data is limited, then curves going through regions with little data will generally be long, and shortest paths, i.e., *geodesics*, will avoid such regions. To ensure that

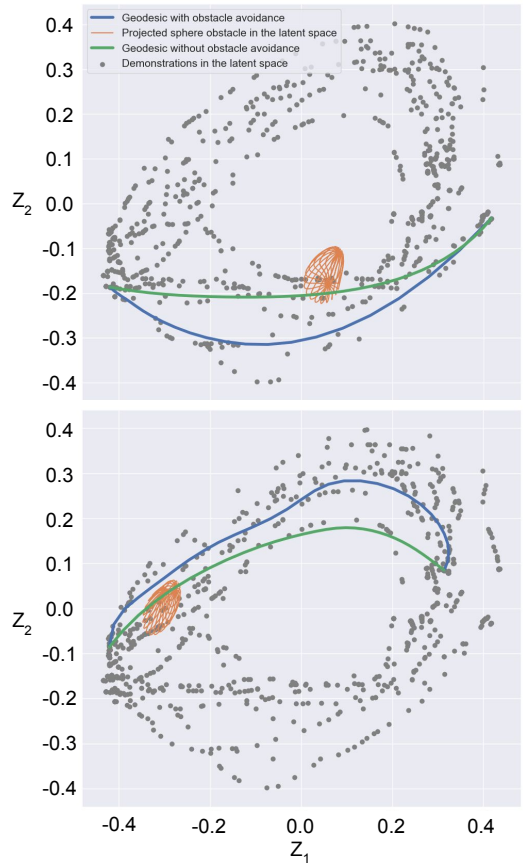

Fig. 2: Geodesics generated in two different scenarios with and without obstacle. The gray points show the training set projected in the latent space. The orange shape is the projection of the wire-frame sphere obstacle.

σ displays this behavior, we follow Arvanitidis *et al.* [4] and use an RBF network for its implementation.

B. Geodesics for motion generation

As geodesics in the latent space tend to follow the trend of data, we suggest using these to generate robot motions that resemble the collected demonstrations. To realize this idea, we train a VAE on a set of robot demonstrations to get a representation that reflects the motion patterns. Note that these demonstrations are collected without any obstacles in place, that is, the robot is not trained with obstacle-avoiding movements.

Obstacle avoidance: Our goal is to eventually allow for dynamic obstacles, which we realize by exploiting a Riemannian metric in ambient space, which displays large distances in regions of obstacles. The resulting geodesic should then avoid these obstacles. For simplicity, we place a Gaussian "bump" at each obstacle

$$m(\boldsymbol{x}) = 1 + 1000 \exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{o}\|^2}{2r^2}\right), \qquad \boldsymbol{x} \in \mathbb{R}^3, \quad (8)$$

where the obstacle has center and radius $(\boldsymbol{o}, r)$. We may interpret this as a Riemannian metric $m(\boldsymbol{x}) \cdot I$.

As we place this metric over the ambient space, we may extend the definition of curve length in latent space to incorporate such a metric [6]

$$\text{Length}[c] = \int_0^1 \sqrt{m\left(f(\boldsymbol{c}_t)\right)} \cdot \|\partial_t f(\boldsymbol{c}_t)\| \mathrm{d}t. \quad (9)$$

This way, the geodesics can adapt to new or dynamic obstacles with no additional learning.

III. EXPERIMENTS

A VAE was trained using the positions $x \in \mathbb{R}^3$ in the task-space demonstrations where the orientations were not considered due to the difficulties caused by approximating unit quaternions using probabilistic methods. Next, the points were shuffled and encoded independently into the latent space $\mathbb{R}^2$. Fig. 1 shows an example of demonstrating a task, where an object is located in the robot workspace. Here, the demonstrations show how to avoid this initial obstacle and reach the point behind it.

The demonstrations were recorded separately, passing by two different sides of the obstacle using a 7-DoF KUKA IIWA and Pyrobolearn [7]. Each of the possible paths was demonstrated 10 times, which makes a total of 20 demonstrations. After training, two points were randomly selected from the test set and the corresponding geodesic between them was generated. To test obstacle avoidance, a new sphere obstacle was added at a random position.

A. Results

Our results show that our approach learns the data manifold, which can be used to generate similar motions by exploiting geodesics generated on the learned Riemannian manifold. As Fig. 1 shows, the RBF network learned to correctly construct a data manifold by assigning high variance to the regions with sparse data and low variance to regions with higher data density. Fig. 2 shows generated geodesics for 2 different scenarios with and without obstacle. The final geodesics correctly avoid the new obstacle meanwhile respecting the geometry of the learned data manifold.

IV. FUTURE WORK

In our future work, we will consider that obstacles can be explicitly modeled and behave dynamically. For more realistic applications, we will extend our approach to handle the end-effector orientations.

REFERENCES

[1] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes.," in *ICLR*, 2014.

[2] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *ICML Wokshop on Unsupervised and Transfer Learning*, 2012.

[3] S. Hauberg, "Only Bayes should learn a manifold (on the estimation of differential geometric structure from data)," in *arXiv preprint*, 2018.

[4] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: on the curvature of deep generative models," *ICLR*, 2018.

[5] D. Eklund and S. Hauberg, "Expected path length on random manifolds," in *arXiv preprint*, 2019.

[6] G. Arvanitidis, S. Hauberg, and B. Schölkopf, "Geometrically enriched latent spaces," in *arXiv preprint*, 2020.

[7] B. Delhaisse, L. Rozo, and D. G. Caldwell, "Pyrobolearn: A Python framework for robot learning practitioners," in *CoRL*, 2019.