IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 September 2023, accepted 2 October 2023, date of publication 6 October 2023, date of current version 13 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3322654

RESEARCH ARTICLE

Geometric Reinforcement Learning for Robotic Manipulation

NASEEM ALHOUSANI[1,2,3], MATTEO SAVERIANO[4], (Senior Member, IEEE), IBRAHIM SEVINC[3], TALHA ABDULKUDDUS[2], HATICE KOSE[1], (Member, IEEE), AND FARES J. ABU-DAKKA[5], (Member, IEEE)

[1]Faculty of Computer and Informatics Engineering, Istanbul Technical University, Maslak, Sarıyer, 80333 İstanbul, Turkey
[2]ILITRON Enerji ve Bilgi Teknolojileri A.Ş, Kağıthane, 34415 İstanbul, Turkey
[3]MCFLY Robot Teknolojileri A.Ş, Sarıyer, 34485 İstanbul, Turkey
[4]Department of Industrial Engineering (DII), University of Trento, 38123 Trento, Italy
[5]Munich Institute of Robotics and Machine Intelligence (MIRMI), Technical University of Munich, 80992 Munich, Germany

Corresponding author: Fares J. Abu-Dakka (fares.abu-dakka@tum.de)

This work was supported in part by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant 3201141, in part by the European Union under NextGenerationEU Project Interconnected Nord-Est Innovation Ecosystem (iNEST) under Grant ECS 00000043, and in part by the European ROBotics and AI Network (euROBIN) Project under Grant 101070596.

ABSTRACT Reinforcement learning (RL) is a popular technique that allows an agent to learn by trial and error while interacting with a dynamic environment. The traditional Reinforcement Learning (RL) approach has been successful in learning and predicting Euclidean robotic manipulation skills such as positions, velocities, and forces. However, in robotics, it is common to encounter non-Euclidean data such as orientation or stiffness, and failing to account for their geometric nature can negatively impact learning accuracy and performance. In this paper, to address this challenge, we propose a novel framework for RL that leverages Riemannian geometry, which we call Geometric Reinforcement Learning ($\mathcal{G}$-RL), to enable agents to learn robotic manipulation skills with non-Euclidean data. Specifically, $\mathcal{G}$-RL utilizes the tangent space in two ways: a tangent space for parameterization and a local tangent space for mapping to a non-Euclidean manifold. The policy is learned in the parameterization tangent space, which remains constant throughout the training. The policy is then transferred to the local tangent space via parallel transport and projected onto the non-Euclidean manifold. The local tangent space changes over time to remain within the neighborhood of the current manifold point, reducing the approximation error. Therefore, by introducing a geometrically grounded pre- and post-processing step into the traditional RL pipeline, our $\mathcal{G}$-RL framework enables several model-free algorithms designed for Euclidean space to learn from non-Euclidean data without modifications. Experimental results, obtained both in simulation and on a real robot, support our hypothesis that $\mathcal{G}$-RL is more accurate and converges to a better solution than approximating non-Euclidean data.

INDEX TERMS Learning on manifolds, policy optimization, policy search, geometric reinforcement learning.

I. INTRODUCTION

Non-Euclidean data, like orientation, stiffness, or manipulability, are important in the field of robotics, as they are widely used during learning and implementation processes [1]. To illustrate, consider real-world scenarios in robotics, like assembly tasks, polishing and grinding, and the automation of industrial welding processes. In such contexts, acquiring knowledge of non-Euclidean data, such as orientation and impedance information, becomes pivotal. Such data have special properties that do not allow the use of Euclidean calculus and algebra. Despite this, they are usually treated as Euclidean data, which demands pre- or post-processing (e.g., normalizing orientation data) to conform to their non-Euclidean nature. This process involves an approximation, and with repetition, the approximation errors will accumulate

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

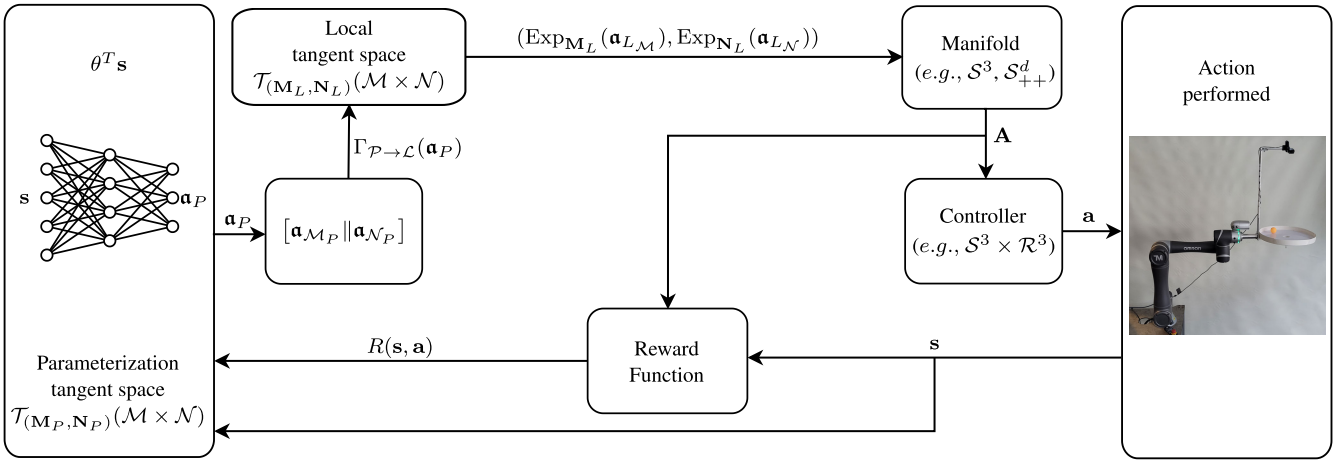


FIGURE 1. Overview of the proposed framework for Geometric Reinforcement Learning ($\mathcal{G}$-RL). Starting with the action from the RL algorithm, the tangent space vector $\mathfrak{a}_P$ is transferred from the parameterization tangent space to the local tangent space through parallel transport. The vector is then mapped onto the corresponding composite manifold to produce the desired action (e.g., orientation and impedance), then passed to the controller to execute the action. The new state and the corresponding action are sent to the reward function to evaluate the quality of the current policy. This evaluation is delivered back to the RL algorithm, along with the full observed state.

until it reaches a level that affects the learning process in terms of accuracy and speed to reach the desired results. This issue was noticed early in the field of statistical learning [2], as determining the center of a non-Euclidean geometric data set using normalization leads to an error in determining the mean. In robotic manipulation, some data belong to different Riemannian manifolds (e.g., 3-sphere manifold $\mathcal{S}^3$ where the unit quaternions live which are possible representations for orientation and Symmetric Positive Definite (SPD) manifold $\mathcal{S}^d_{++}$ for stiffness and manipulability), and proper mathematical tools need to be developed in order to avoid approximations [3].

Nowadays, the practical applications of RL have become many and span various fields [4], [5], [6] including robotics [7]. In RL, a policy produces actions based on the current state. When these actions and/or states have a geometric meaning (manifold data) like orientation, stiffness, or manipulability, it requires pre- and post-processing to preserve and benefit from the data geometry; neglecting the underlying constraints of the manifold of these data leads to inaccuracy in both exploration and learning. This is valid for deterministic and probabilistic RL approaches. Nevertheless, a distribution is learned for probabilistic RL algorithms instead of a single action. This adds more challenges when it samples manifold data from a distribution like the Gaussian distribution. For both cases, benefiting from Riemannian geometry can potentially improve the quality of learned policies.

In this paper, we propose a novel RL (Geometric Reinforcement Learning ($\mathcal{G}$-RL)) framework leveraging Riemannian geometry to exploit the geometric structure of the robotic manipulation data. This framework involves applying policy parameterization on the tangent space of a base point on the manifold, followed by using parallel transport to transport the action in a tangent space that moves with the active point. The result is then mapped to its corresponding non-Euclidean manifold. We apply $\mathcal{G}$-RL to learn and predict actions, like orientation data represented as unit quaternions or stiffness and manipulability data encapsulated in SPD matrices. The proposed $\mathcal{G}$-RL framework has been applied to extend two prominent deep reinforcement learning algorithms—Soft Actor-Critic (SAC) [8] and Proximal Policy Optimization (PPO) [9])—to work with manifold data. Furthermore, we have applied it to Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10] (which belongs to the family of Black Box Optimization (BBO) algorithms). CMA-ES can be used as a policy improvement method like in [11]. An overview of $\mathcal{G}$-RL is shown in Fig. 1.

To summarize, our work can be outlined by the following contributions:

- A novel geometry-aware RL framework, namely $\mathcal{G}$-RL, that incorporates Riemannian geometry to enable agents to learn robotic manipulation skills with non-Euclidean data.
- Different instantiations of $\mathcal{G}$-RL to extend popular model-free RL approaches including:
  - model-free RL algorithms (e.g., Policy learning by Weighting Exploration with the Returns (PoWER)),
  - model-free deep RL algorithms (e.g., SAC and PPO), and
  - BBO algorithms (e.g., CMA-ES).
- Extensive evaluation and experimentation with simulations and a physical robot, with comparisons to different distributions and baselines.

The rest of the paper is organized as follows. Section II discusses related work; Section III provides a short background about RL and Riemannian manifolds; Section IV presents our proposed approach; Section V shows experimental results from both simulation and a physical robot; Section VI

discusses the results and the limits of our approach; and we conclude the paper in Section VII.

II. RELATED WORK

Most conventional RL algorithms that utilize a Gaussian distribution (e.g., [8], [9], [12], [13], [14], [15], [16], [17]) are not suitable for accurately learning non-Euclidean data. This is because such data resides in a curved space rather than a vector space and therefore requires a special treatment to avoid approximation errors and to account for its unique properties of that data. This has been approached in different ways.

The Riemannian manifold, Riemannian metric, and tangent space are mathematical concepts utilized in creating geometrical tools for statistics on manifolds as in [2], which have been used in different research works. Abu-Dakka et al. [1] leveraged Riemannian geometry in the context of learning robotic manipulation skills (e.g., stiffness, manipulability, and covariance) using a kernelized treatment in the tangent space. Huang et al. [18] proposed adapting learned orientation trajectories to pass through via-points or end-points while also considering the angular velocity. Work done in [19] utilizes a variational autoencoder (VAE) to learn geodesics on Riemannian manifolds using Learning from Demonstration (LfD), which generates end-effector pose trajectories able to dynamically avoid obstacles present in the environment.

In contact-rich manipulation tasks, it is not safe to only use position control. Research in [20] combines contact Dynamic Movement Primitivess (DMPs) with SAC to adapt impedance and learn both linear and orientation stiffness according to a given force and position trajectories, which is then passed into an adaptive admittance controller for robotic manipulation. However, stiffness in their research is represented as a diagonal matrix, whereas our approach can learn the full stiffness matrix. Representing stiffness as a diagonal matrix avoids computational complexity on account of accuracy. But in some cases, such as examining stability properties, it is important to note that the off-diagonal elements of the stiffness matrix can have a direct impact. Disregarding these elements may result in an imprecise assessment of stability [21]. Additionally, the off-diagonal elements in the stiffness matrix correspond to the interaction between various degrees of freedom. If these elements are ignored by employing a diagonal matrix, it can result in an oversimplified analysis, causing the loss of vital information. Off-diagonal interactions can occur due to physical connections, interdependencies among variables, or constraints within the system. Utilizing the full stiffness matrix allows researchers to precisely account for and assess these interactions.

Authors of [22] also employ a diagonal stiffness matrix in the context of variable impedance, using Inverse Reinforcement Learning (IRL) to discover the reward function in addition to the policy from an expert demonstration (LfD). They proposed that their algorithm can be extended to the full stiffness matrix using Cholesky decomposition. We used Cholesky decomposition as a baseline, and our results show that our algorithm outperforms this baseline.

In [23], authors studied the use of LfD for force sensing and variable impedance control, with the proposed framework able to use both Cholesky decomposition and Riemannian manifold representations of stiffness. The main difference from our work is that they did not use RL; their work was instead based on LfD.

Although researchers in [24] proposed a method for online selection of non-diagonal stiffness matrices for admittance control using RL, they still learn to select from a few previously defined full stiffness matrices. Our algorithm can learn the full stiffness matrix online.

In the context of image segmentation, authors in [25] proposed a method for 3D image reconstruction. They achieved this by modifying the original CMA-ES to work on Riemannian manifolds and applying optimization on the tangent space. Unlike our approach of optimizing in the parameter space where geometric data is parameterized using Euclidean parameters, their technique optimizes geometric data directly. In computer vision, parameterization on the tangent space is commonly used to regress rotations with deep learning, as explained in [26].

In [27], the utilization of Riemannian manifolds with a solitary optimized tangent space was employed to ensure compliance of parameterization results with manifold geometry. Our work, in contrast, proposes the utilization of two tangent spaces: one for parameterization and another for mapping in the exploration neighborhood. Specifically, our approach maintains proximity of the local tangent space to the active exploration region of the manifold, resulting in more effective utilization of the Riemannian geometry.

The authors of [28] proposed a policy equivariant to $\mathcal{SO}(2)$ when the reward and transition functions are invariant to that group. This work is interesting and makes the learning of the elements of $\mathcal{SO}(2)$ faster. However, it does not discuss how to treat orientation data while learning the policy. Our work learns a policy (i.e., orientation) while considering the geometry of manifold data.

Researchers have explored the application of optimization algorithms on Riemannian manifolds. The authors of [29] employedBayesian Optimization (BO) to optimize policy parameters and introduced geometry-aware kernels. These kernels enable proper measurement of the similarity between Riemannian manifold parameters using Gaussian process (GP). Another recent work, in [30], implemented the geometry-aware Riemannian Matérn kernels in the domain of robotics. These investigations consider non-Euclidean manifolds' geometry and propose a geometry-aware framework. Given the advantages of Riemannian geometry in BO, we endeavor to exploit it in the realm of RL.

Policy learning in $\mathcal{SE}(3)$ actions is proposed in [31], achieved by factorizing high dimensional action spaces into several smaller action spaces with progressively augmented state spaces. Each action space is handled by its own neural

network. This work is primarily focused on learning poses by imitation of images. A limitation of this work is its use of Euler angles to represent the orientations and being restricted to ±30 degrees rotations out of the plane. The geometry of the orientation data is also not considered, as there is no explanation or discussion about it in the paper.

Although there are many existing works in the field of LfD and supervised learning, Riemannian geometry has not been exploited in RL. A recent work, Bingham Policy Parameterization (BPP) [32], uses the Bingham distribution as an alternative to the Gaussian distribution for learning orientation policies. This choice was motivated by the argument that unit quaternions can be directly sampled from the Bingham distribution, unlike the Gaussian distribution, where one must use normalization. Nevertheless, authors in [32] reported that as their implementation uses several neural networks, instability in the learning process could occur if erroneous data is sampled from them. Furthermore, our algorithm is not limited to a special distribution such as the Bingham distribution, which is constrained to the sphere manifold. As a result, it can effectively handle data from other types of manifolds, such as $\mathcal{S}^d_{++}$. We experimentally compare the performance of $\mathcal{G}$-RL and BPP in Sec. V-A1.

III. BACKGROUND

A. REINFORCEMENT LEARNING

The general formulation of a typical RL problem is about an agent at time t in state $\mathbf{s}_t$ selecting an action $\mathbf{a}_t$ according to a stochastic policy

$$\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}) = \Pr(\mathbf{a} = \mathbf{a}_t \mid \mathbf{s} = \mathbf{s}_t), \tag{1}$$

where $\boldsymbol{\theta} \in \mathcal{R}^n$ are the parameters of the policy and π is the probability distribution of sampling action $\mathbf{a}_t$ in state $\mathbf{s}_t$ at time t. Performing action $\mathbf{a}_t$ changes the world state to $\mathbf{s}_{t+1}$ and the agent receives a reward r_{t+1}, associated with the transition $T(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. The agent's objective is to maximize the expected return of the policy [33], i.e.,

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R(\mathbf{s}, \mathbf{a})] = \max_{\boldsymbol{\theta}} \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]. \tag{2}$$

In this paper, we have used different RL algorithms and a BBO algorithm for policy improvement to show the versatility of our proposed approach. The used algorithms are briefly reviewed as follows.

1) POWER

PoWER [13] is an RL policy search algorithm inspired by expectation maximization in supervised learning algorithms. It is designed for finite horizons with episodic restarts and uses an average return as a weight instead of a gradient.

2) SAC

SAC [8] is an instance of entropy-regularized deep RL, which aims to maximize the policy's return while also maximizing entropy. An entropy coefficient is used to control the importance of entropy and is adjusted during training.

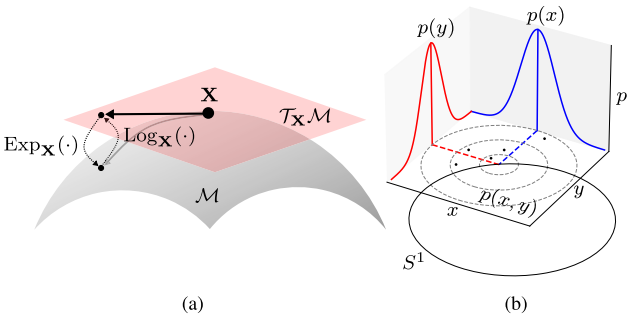


FIGURE 2. (a) The gray surface represents a manifold $\mathcal{M}$, and the red plane represents the tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$. The exponential/logarithmic mapping tools between the two spaces are shown. (b) Sampling an S^1 manifold from a Gaussian distribution, where the mean is on S^1, but the drawn samples may not be, like the points in $p(x, y)$.

3) PPO

PPO [9] is a deep RL policy gradient optimization algorithm that clips policy gradient updates to a narrow interval, ensuring the new policy is not too far from the existing one.

4) CMA-ES

CMA-ES [10] is a derivative-free method for non-linear or non-convex BBO problems in the continuous domain. Instead of using gradient information, CMA-ES makes use of evolutionary computation and an evolution strategy to solve the optimization problem.

B. RIEMANNIAN MANIFOLD

A Riemannian manifold $\mathcal{M}$ is an n-dimensional smooth differentiable topological space equipped with a Riemannian metric that locally resembles the Euclidean space $\mathcal{R}^n$. The locally Euclidean tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ can be constructed around any point $\mathbf{X} \in \mathcal{M}$. The Riemannian metric, defined as the positive definite inner product, can be used to generalize the notion of the straight line between two points in Euclidean space by defining the shortest curve between two points in a manifold, which is denoted as a geodesic.

In order to go back and forth between a manifold $\mathcal{M}$ and a tangent space $\mathcal{T}_{\mathbf{X}}\mathcal{M}$, we require two distance-preserving mapping functions (operators). These operators are (*i*) the exponential map $\mathrm{Exp}_{\mathbf{X}} : \mathcal{T}_{\mathbf{X}}\mathcal{M} \rightarrow \mathcal{M}$, and its inverse (*ii*) the logarithmic map $\mathrm{Log}_{\mathbf{X}} : \mathcal{M} \rightarrow \mathcal{T}_{\mathbf{X}}\mathcal{M}$ as depicted in Fig. 2 (a). It is possible to show that exponential and logarithmic maps are (locally) bijective [34], which makes it possible to do the calculations about the non-Euclidean manifold space on the tangent space and project back the results. Another essential concept in differential geometry is parallel transport $\Gamma_{\mathbf{X}\rightarrow\mathbf{Y}}$, allowing for calculations and comparisons of vectors located on different tangent spaces to be carried out by moving vectors through a connecting geodesic. This method preserves the inner product between transported vectors.

A Gaussian distribution on Riemannian manifolds is defined as in [3]

$$\mathcal{N}_{\mathcal{M}}(\mathbf{Q}|\mathbf{X}, \mathbf{\Sigma}) = \left((2\pi)^d |\mathbf{\Sigma}|\right)^{-\frac{1}{2}} e^{\text{Log}_{\mathbf{X}}(\mathbf{Q})\mathbf{\Sigma}^{-1}\text{Log}_{\mathbf{X}}(\mathbf{Q})}, \quad (3)$$

where $\mathbf{X} \in \mathcal{M}$, the covariance $\mathbf{\Sigma}$ is defined on $\mathcal{T}_{\mathbf{X}}\mathcal{M}$ and $\mathbf{Q} \in \mathcal{M}$. For more details about Gaussian distributions on manifolds in the context of robotics, we refer the interested reader to [3].

IV. POLICY PARAMETERIZATION ON TANGENT SPACE

Recently, the topic of learning using Riemannian geometry tools has become the focus of researchers in the field of robot learning [1], [3], [35], [36]. An example of this is when considering a robot's operational space, where its end-effector pose consists of a Cartesian position (Euclidean part) and orientation (non-Euclidean part). It is common to apply learning in this space since it allows for kinematic redundancy and the ability to transfer a learned policy from one robot to another robot with different anatomy [37].

Gaussian policy parameterization has a limitation when it comes to representing non-Euclidean data like orientation, stiffness, or manipulability, as the distribution parameters (both mean and covariance) do not always obey the nature of the manifold's curvature space. The problem with sampling non-Euclidean from a Gaussian distribution is illustrated in Fig. 2 (b) for the S^1 manifold, i.e., the circumference of the unit circle. Picking a point on the manifold to be the mean of the normal distribution, samples can still be drawn from outside the manifold, as illustrated in the figure. Normalization of the sample can map it back to the unit circle manifold at the cost of accuracy. The same argument is applicable to other manifolds like $\mathcal{S}^3$ embedded in $\mathcal{R}^4$. To this point, using Gaussian policy parameterization like SAC [8] or PPO [9] on non-Euclidean manifold data like quaternions will require normalizing the predicted profiles. This kind of post-processing is an approximation that could affect learning accuracy.

Having a framework allowing for well-established and stable learning algorithms on Euclidean space to be transferred to other geometrical spaces with relative ease and reasonable computational costs is beneficial. This enables part of the achievements and progress that have been made on Euclidean space to be directly applicable to non-Euclidean spaces. $\mathcal{G}$-RL is based on applying parameterization on a constant tangent space, where there is no need to parallel transport the policy being learned from one tangent space to another. Doing so is not trivial for some parameterization schemes. At the same time, we must obey the formulation of the Riemannian geometry, which is locally bijective.

Thus, let us consider $\mathcal{M}$ and $\mathcal{N}$ as two Riemannian manifolds, where $\mathbf{M}_P, \mathbf{M}_L \in \mathcal{M}$ and $\mathbf{N}_P, \mathbf{N}_L \in \mathcal{N}$. Conceptually, when $\mathcal{G}$-RL is used to learn data that correspond to a Riemannian manifold $\mathcal{M}$, we utilize the tangent space in two ways: a constant tangent space $\mathcal{T}_{\mathbf{M}_P}\mathcal{M}$ for parameterization, and a local tangent space $\mathcal{T}_{\mathbf{M}_L}\mathcal{M}$ for mapping to manifold actions. The manifolds' data which are indexed with P represent the data points where the parameterization tangent spaces are established, and the ones indexed with L represent the data points where the moving local tangent spaces are established.

In the case where consecutive actions i and $i+1$ are local to one another (such as learning a smooth trajectory of orientations), the local tangent space is situated on the previous action (e.g., predicting the orientation at time $i+1$ means situating the local tangent space on the predicted orientation at time i). Parallel transport must then be employed to move the parameterized vectors to the local tangent spaces. But in the case where the rollout consists of a single action and the different rollouts are independent of each other (e.g., Wahba problem), we locate both the parameterization tangent space and local tangent space onto the same point. Note that the parameterization tangent space is never moved itself; the policy is learned on a fixed, constant tangent space. In either case, we map the result back to the manifold once the vector is moved to the local tangent space.

In the general setting of learning a manipulation task, it is common to have state and action data from different manifolds, in other words, having a composite manifold, which is defined as the Cartesian product of the manifolds. For example, the state $\mathbf{s} \in \mathcal{M} \times \mathcal{N}$ and the action $\mathbf{a} \in \mathcal{M} \times \mathcal{N}$.

The parameterization tangent space of the composite manifold is represented as $\mathcal{P}$: $\mathcal{T}_{(\mathbf{M}_P,\mathbf{N}_P)}(\mathcal{M} \times \mathcal{N})$, while the local tangent space of the composite manifold is represented by $\mathcal{L}$: $\mathcal{T}_{(\mathbf{M}_L,\mathbf{N}_L)}(\mathcal{M} \times \mathcal{N})$.

The state at time t as a composite manifold state is represented as follows:

$$\mathbf{s}_t = (\mathbf{S}_{\mathcal{M},t}, \mathbf{S}_{\mathcal{N},t}), \quad (4)$$

where $\mathbf{S}_{\mathcal{M},t}$ and $\mathbf{S}_{\mathcal{N},t}$ are the state parts that belong to each of the two manifolds $\mathcal{M}$ and $\mathcal{N}$, respectively. The action $\mathfrak{a}_{P,t}$ on the composite parameterization tangent space at time t is represented as follows:

$$\mathfrak{a}_{P,t} = [\mathfrak{a}_{P_{\mathcal{M}},t} \| \mathfrak{a}_{P_{\mathcal{N}},t}], \quad (5)$$

while the action $\mathfrak{a}_{L,t}$ on the composite local tangent space is given by

$$\mathfrak{a}_{L,t} = [\mathfrak{a}_{L_{\mathcal{M}},t} \| \mathfrak{a}_{L_{\mathcal{N}},t}], \quad (6)$$

where the subscripts $_{\mathcal{M},t}$ and $_{\mathcal{N},t}$ denote the part of the action coming from manifolds $\mathcal{M}$ and $\mathcal{N}$, respectively. The $[\cdot\|\cdot]$ is a concatenation operator. Intuitively, the prediction on the tangent space allows us to "stack" different manifolds into a unique vector. Afterward, we use the parallel transport operator to transport the action vector from $\mathcal{P}$ to $\mathcal{L}$ at t as

$$\mathfrak{a}_{L,t} = \Gamma_{\mathcal{P}\to\mathcal{L}}(\mathfrak{a}_{P,t}). \quad (7)$$

Subsequently, we project this local action vector to the composite manifold as follows

$$\begin{aligned}\mathbf{a}_t &= (\mathbf{A}_{\mathcal{M},t}, \mathbf{A}_{\mathcal{N},t}) \\ &= (\text{Exp}_{\mathbf{M}_{L,t}}(\mathfrak{a}_{L_{\mathcal{M}},t}), \text{Exp}_{\mathbf{N}_{L,t}}(\mathfrak{a}_{L_{\mathcal{N}},t})).\end{aligned} \quad (8)$$

Algorithm 1 Geometric Reinforcement Learning ($\mathcal{G}$-RL)

Input: initial state $\mathbf{s}_0$, initial parameters $\boldsymbol{\theta}$, $\mathbf{M}_P, \mathbf{M}_0 \in \mathcal{M}$, $\mathbf{N}_P, \mathbf{N}_0 \in \mathcal{N}$, where $\mathbf{M}_P$, and $\mathbf{N}_P$ are the centers of the composite parameterization tangent space. $\mathbf{M}_0$, $\mathbf{N}_0$ are the centers of the initial composite local tangent space and the RL algorithm α.

1: **while** ! `stop_condition(`α`)` **do**
2: $\quad\pi_{\boldsymbol{\theta}}(\mathfrak{a}_P|\mathbf{s}) \leftarrow$ `get_policy(`$\boldsymbol{\theta}, \alpha$`)` ▷ eq. (9)
3: $\quad R(\mathbf{s}, \mathbf{a}) \leftarrow 0$ ▷ cumulative reward
4: $\quad$**for** $t = 0, \ldots, T-1$ **do**
5: $\quad\quad\mathbf{s}_t \leftarrow (\mathbf{S}_{\mathcal{M},t}, \mathbf{S}_{\mathcal{N},t})$ ▷ state composition (4)
6: $\quad\quad\mathfrak{a}_{\mathcal{P},t} \sim \pi_{\boldsymbol{\theta}}(\mathfrak{a}_{\mathcal{P},t}|\mathbf{s}_t)$ ▷ tangent space action (10)
7: $\quad\quad\mathfrak{a}_{\mathcal{P},t} = \left[\mathfrak{a}_{P_{\mathcal{M}},t} \| \mathfrak{a}_{P_{\mathcal{N}},t}\right]$ ▷ act. concatenation (5)
8: $\quad\quad\mathfrak{a}_{\mathcal{L},t} \leftarrow \Gamma_{\mathcal{P}\rightarrow\mathcal{L}_t}(\mathfrak{a}_{\mathcal{P},t})$ ▷ act. par. trans. (7)
9: $\quad\quad\mathfrak{a}_{\mathcal{L},t} = \left[\mathfrak{a}_{L_{\mathcal{M}},t} \| \mathfrak{a}_{L_{\mathcal{N}},t}\right]$ ▷ act. concatenation (6)
10: $\quad\quad\mathbf{a}_t \leftarrow (\mathrm{Exp}_{\mathbf{M}_{L,t}}(\mathfrak{a}_{L_{\mathcal{M}},t}), \mathrm{Exp}_{\mathbf{N}_{L,t}}(\mathfrak{a}_{L_{\mathcal{N}},t}))$ ▷ manifold act. (8)
11: $\quad\quad(\mathbf{M}_{t+1}, \mathbf{N}_{t+1}) \leftarrow (\mathbf{S}_{\mathcal{M},t}, \mathbf{S}_{\mathcal{N},t})$
12: $\quad\quad\mathbf{s}_{t+1} \leftarrow$ `execute_on_robot(`$\mathbf{a}_t$`)`
13: $\quad\quad R(\mathbf{s}, \mathbf{a}) \leftarrow R(\mathbf{s}, \mathbf{a}) + r(\mathbf{s}_t, \mathbf{a}_t)$
14: $\quad$**end for**
15: $\quad\boldsymbol{\theta} \leftarrow$ `improve_policy(`$\boldsymbol{\theta}, R(\mathbf{s}, \mathbf{a}), \alpha$`)`
16: **end while**

The policy $\pi_{\boldsymbol{\theta}}$ predicts the action on the parameterization tangent space $\mathfrak{a}_P$ according to the current state $\mathbf{s}$ as follows:

$$\pi_{\boldsymbol{\theta}}(\mathfrak{a}_P|\mathbf{s}) = \left[\pi_{\boldsymbol{\theta}_{\mathcal{M}}}\left(\mathfrak{a}_{P_{\mathcal{M}}}|\mathbf{S}_{\mathcal{M}}\right) \| \pi_{\boldsymbol{\theta}_{\mathcal{N}}}(\mathfrak{a}_{P_{\mathcal{N}}}|\mathbf{S}_{\mathcal{N}})\right] \quad (9)$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\mathcal{M}} \| \boldsymbol{\theta}_{\mathcal{N}}]$ is the concatenation of parameters for the manifolds, respectively. At each time t, an action $\mathfrak{a}_{P,t} = [\mathfrak{a}_{P_{\mathcal{M}},t} \| \mathfrak{a}_{P_{\mathcal{N}},t}]$ is drawn from the policy (9) as

$$\mathfrak{a}_{P,t} \sim \left[\pi_{\boldsymbol{\theta}_{\mathcal{M}}}\left(\mathfrak{a}_{P_{\mathcal{M}},t}|\mathbf{S}_{\mathcal{M},t}\right) \| \pi_{\boldsymbol{\theta}_{\mathcal{N}}}(\mathfrak{a}_{P_{\mathcal{N}},t}|\mathbf{S}_{\mathcal{N},t})\right]. \quad (10)$$

The action $\mathfrak{a}_{P,t}$ is converted in a manifold action $\mathbf{a}_t$ using (8), and the agent performs the resulting manifold action on the environment. This causes the state to transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. The expected return captures the expected quality of the policy

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_t r((\mathbf{S}_{\mathcal{M},t}, \mathbf{S}_{\mathcal{N},t}), (\mathbf{A}_{\mathcal{M},t}, \mathbf{A}_{\mathcal{N},t}))\right]. \quad (11)$$

As shown in Algorithm 1, the initial state, the centers of the composite of two tangent spaces, and the RL algorithm are used as input. In line 2, the RL algorithm generates a policy structure $\pi_{\boldsymbol{\theta}}(\mathfrak{a}|\mathbf{s})$ with the current parameters $\boldsymbol{\theta}$. This policy operates in the composite parameterization tangent space established at the composite point $(\mathbf{M}_P, \mathbf{N}_P)$ given as input to the RL algorithm. The parameterization is based on the composite current state in line 5, as it is passed to the policy in line 6 to sample the composite action $\mathfrak{a}_P$ on the parameterization tangent space. This action (defined in line 7) is parallel transported to the current local composite tangent space (line 8) and gives the action vector defined in line 9. Line 10 maps the composite tangent space action into the composite manifold. After that, the local composite tangent space is updated to the current composite state (line 11), the action is executed by the agent, and the state is updated (line 12). At each step in the rollout, the total reward is updated by accumulating the immediate rewards (line 13). After one rollout is finished, the quality of the policy is measured using the rollout total reward, which is passed to the RL algorithm to proceed with learning (line 15). This procedure is repeated until the stopping criteria, depending on the used RL algorithm, is met (line 1).

A. LEARNING ON THE $\mathcal{S}^3$ MANIFOLD

Orientations are commonly represented using rotation matrices, Euler angles, or unit quaternions. Euler angles are a minimal orientation representation (requiring only three parameters) but suffer from the singularity problem [38]. Unit quaternions hold an advantage over rotation matrices due to requiring fewer parameters (4 instead of 9) and are therefore commonly used to represent rotation in robotic applications. The unit quaternion representation belongs to the 3-sphere manifold, denoted as $\mathcal{S}^3$ [38]. Therefore, applying current reinforcement learning algorithms designed for Euclidean space to learn an orientation policy is not straightforward as it normally involves approximations to account for the underlying manifold structure.

In this section, we focus on orientation learning represented by unit quaternions. A quaternion, denoted as $\mathbf{Q}$, is a tuple $(v, \mathbf{u})$ composed of a scalar v and a three-dimensional vector $\mathbf{u} = (x, y, z)$. Unit quaternions have a norm of one and belong to $\mathcal{S}^3$. The hypersphere $\mathcal{S}^3$ has a double-covering of $\mathcal{SO}(3)$, meaning that for every rotation in $\mathcal{SO}(3)$ there exist two quaternions that can represent it ($\mathbf{Q}$ and $-\mathbf{Q}$). In this section, actions can be represented as unit quaternions, and the learning is carried out on a single hemisphere; in case we have a prediction $\mathbf{Q}$ on the other hemisphere, we flip the prediction by using $-\mathbf{Q}$.

In order to effectively utilize Gaussian distribution calculations for unit quaternions, it is necessary to take into account their geometric properties. The objective is to maximize the expected reward as defined in equation (11).

In this context, we define $\mathcal{M} \equiv \mathcal{S}^3$, and consider unit quaternions $\mathbf{Q} = (v, \mathbf{u})$, $\mathbf{Q}_1 = (v_1, \mathbf{u}_1)$, $\mathbf{Q}_2 = (v_2, \mathbf{u}_2) \in \mathcal{S}^3$, and $\mathfrak{q}, \mathfrak{u} \in \mathcal{T}_{\mathbf{Q}}\mathcal{S}^3$ The logarithmic map, denoted as $\mathrm{Log}_{\mathbf{Q}_1}(\cdot)$ is redefined to map $\mathbf{Q}_2$ into $\mathcal{T}_{\mathbf{Q}_1}\mathcal{S}^3$ e.g., $\mathrm{Log}_{\mathbf{Q}_1}(\cdot) : \mathcal{S}^3 \mapsto \mathcal{R}^4$ [39] as

$$\mathrm{Log}_{\mathbf{Q}_1}(\mathbf{Q}_2) = \frac{\mathbf{Q}_2 - (\mathbf{Q}_1{}^\top \mathbf{Q}_2)\mathbf{Q}_1}{\|\mathbf{Q}_2 - (\mathbf{Q}_1{}^\top \mathbf{Q}_2)\mathbf{Q}_1\|} d(\mathbf{Q}_1, \mathbf{Q}_2), \quad (12)$$

where $\|\cdot\|$ defines the norm of a vector, and the distance between two unit quaternions is defined as follows

$$d(\mathbf{Q}_1, \mathbf{Q}_2) = \arccos(\mathbf{Q}_1{}^\top \mathbf{Q}_2), \quad (13)$$

For example, if the reward function is $\exp(-d)$, where d is the distance between two unit quaternions, then equation (13) is used to calculate the distance on the tangent space.

In (8), the exponential map, denoted as $\text{Exp}_{\mathbf{Q}_1}(\cdot)$, is redefined to project actions from the current local tangent space into the hypersphere $\mathcal{S}^3$, e.g., $\text{Exp}_{\mathbf{Q}_1}(\cdot) : \mathcal{R}^4 \mapsto \mathcal{S}^3$ [39]

$$\text{Exp}_{\mathbf{Q}_1}(\mathbf{q}) = \mathbf{Q}_1 \cos(\|\mathbf{q}\|) + \frac{\mathbf{q}}{\|\mathbf{q}\|} \sin(\|\mathbf{q}\|). \tag{14}$$

Parallel transport in (7) is redefined as in [39]:

$$\Gamma_{\mathbf{Q}_1 \to Q_2}(\mathbf{q}) = \left(-\mathbf{Q}_1 \sin(\|\mathbf{u}\|)\overline{\mathbf{u}}^\top + \overline{\mathbf{u}} \cos(\|\mathbf{u}\|)\overline{\mathbf{u}}^\top + \left(\mathbf{I} - \overline{\mathbf{u}}\overline{\mathbf{u}}^\top\right)\right) \mathbf{q} \tag{15}$$

with $\overline{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}$, and $\mathbf{u} = \text{Log}_{\mathbf{Q}_1}(\mathbf{Q}_2)$.

B. LEARNING ON THE $\mathcal{S}^d_{++}$ MANIFOLD

Data such as stiffness, manipulability, and covariance ellipsoids/matrices play a vital role in robotic manipulation. Such data belong to the space of SPD matrices. However, effectively learning these data using RL algorithms is challenging due to the need for approximations to conform to the manifold geometry. Typically, Cholesky decomposition is employed to guarantee that the predicted matrix remains SPD [23].

A matrix $\mathbf{\Sigma}$ belongs to the space $\mathcal{S}^d_{++}$ if it satisfies two conditions: symmetry (i.e., $\mathbf{\Sigma} = \mathbf{\Sigma}^\top$) and positive definiteness $\mathbf{v}^\top \mathbf{\Sigma} \mathbf{v} > 0$, $\forall$ nonzero vectors $\mathbf{v}$. To express manifold operators for SPD matrices as outlined in [40] and [41], we introduce the notation $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2, \mathbf{W} \in \mathcal{S}^d_{++}$ and $\mathfrak{w} \in \mathcal{T}_{\mathbf{\Sigma}}\mathcal{S}^d_{++}$. The exponential map, denoted as $\text{Exp}_{\mathbf{\Sigma}}(\cdot)$ in (8), is redefined to project actions from the current local tangent space to the $\mathcal{S}^d_{++}$ manifold

$$\text{Exp}_{\mathbf{\Sigma}}(\mathfrak{w}) = \mathbf{\Sigma}^{\frac{1}{2}} \text{expm}\left(\mathbf{\Sigma}^{-\frac{1}{2}} \mathfrak{w} \mathbf{\Sigma}^{-\frac{1}{2}}\right) \mathbf{\Sigma}^{\frac{1}{2}} \tag{16}$$

The logarithmic map, denoted as $\text{Log}_{\mathbf{\Sigma}}(\cdot)$, is redefined to map $\mathbf{W}$ to $\mathcal{T}_{\mathbf{\Sigma}}\mathcal{S}^d_{++}$

$$\text{Log}_{\mathbf{\Sigma}}(\mathbf{W}) = \mathbf{\Sigma}^{\frac{1}{2}} \text{logm}\left(\mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{W} \mathbf{\Sigma}^{-\frac{1}{2}}\right) \mathbf{\Sigma}^{\frac{1}{2}} \tag{17}$$

Parallel transport is defined as

$$\mathcal{T}_{\mathbf{\Sigma}_1 \to \mathbf{\Sigma}_2}(\mathfrak{w}) = \mathbf{\Sigma}_2^{\frac{1}{2}} \mathbf{\Sigma}_1^{-\frac{1}{2}} \mathfrak{w}\ \mathbf{\Sigma}_1^{-\frac{1}{2}\top} \mathbf{\Sigma}_2^{\frac{1}{2}\top} \tag{18}$$

The distance between two SPD matrices is defined as follows

$$d(\mathbf{\Sigma}, \mathbf{W}) = \left|\left|\text{logm}\left(\mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{W} \mathbf{\Sigma}^{-\frac{1}{2}}\right)\right|\right|_F \tag{19}$$

where $||\cdot||_F$ is the Frobenius norm.

An important feature of $\mathcal{S}^d_{++}$ is that it has no cut locus, resulting in a bijective mapping over the entire manifold space [40].

When parameterizing SPD matrices, two approaches were used: vectorization via both Cholesky factorization and Mandel notation. In the Cholesky factorization approach, an SPD matrix Σ is represented as the product of its Cholesky factor $\mathbf{L}$ and its transpose, i.e., $\Sigma = \mathbf{L}^\top \mathbf{L}$. The vectorization is then performed on the upper triangle elements of $\mathbf{L}$ for learning purposes. Alternatively, in the Mandel notation approach, an SPD matrix Σ can be defined using a specific vector representation. For example, in the case of 3×3 SPD matrix $[\Sigma] = [\Sigma_{11}, \Sigma_{22}, \Sigma_{33}, \sqrt{2}\Sigma_{23}, \sqrt{2}\Sigma_{13}, \sqrt{2}\Sigma_{12}]^\top$. In $\mathcal{G}$-RL implementation for SPD data, we utilized the Mandel notation to reduce the dimensionality of the data. Additionally, we used Mandel notation as a baseline, referred to as "Mandel," where we find the nearest SPD matrix to the predicted symmetric matrix. We experimentally evaluate both vectorization approaches in Section V-A2 and Section V-A4.

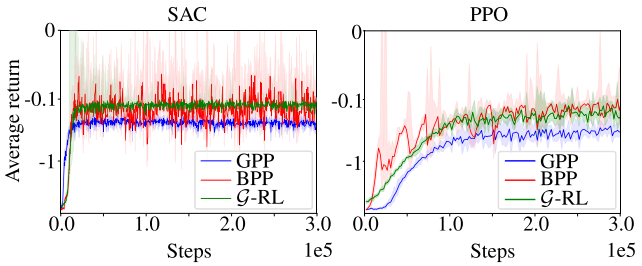


FIGURE 3. Quaternion Wahba domain results for SAC and PPO using Gaussian Policy Parameterization (GPP), $\mathcal{G}$-RL, and BPP. The mean (solid lines) and the standard deviation (shaded regions) are calculated over five different seeds.

V. EXPERIMENTAL RESULTS

Experiments have been carried out in simulated environments (Wahba [42] and trajectory learning problems), as well as a real setup involving a physical robot performing the Ball-in-a-hole task. Several RL and policy improvement algorithms have been tested:

- deep RL algorithms like SAC [8] and PPO [9],
- the expectation-maximization inspired PoWER algorithm [13], and
- the BBO-based CMA-ES algorithm [10].

Our research question is about the gains of considering the geometry of non-Euclidean data (e.g., orientation, stiffness, or manipulability) in RL algorithms based on Gaussian distributions and how they compare with the common approximation solutions (e.g., normalization and Cholesky decomposition) or solutions based on other distributions like Bingham.

A. SIMULATION EXPERIMENTS

1) QUATERNION WAHBA PROBLEM

The Wahba problem, first proposed by Grace Wahba in 1965 [42], is about finding the best rotation between two Euclidean coordinate systems that aligns two sets of noisy 3-dimensional vector observations. The original motivation for this problem was to estimate satellite altitudes using vectors from different frames of reference, but it was later applied to other research fields as well.

The cost function defines attempts to minimize the difference between sets of vectors ($\mathbf{y}_i \in Y, \mathbf{z}_i \in Z$) by finding

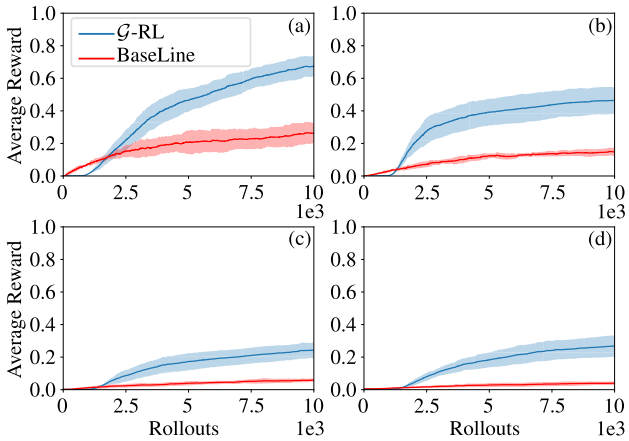


FIGURE 4. Four instances of the variation of the Wahba problem with different sizes (complexity) solved by the PoWER algorithm. The size for each is (a) 10, (b) 12, (c) 14, and (d) 16. The mean (solid lines) and the standard deviation (shaded regions) are calculated over five different seeds.

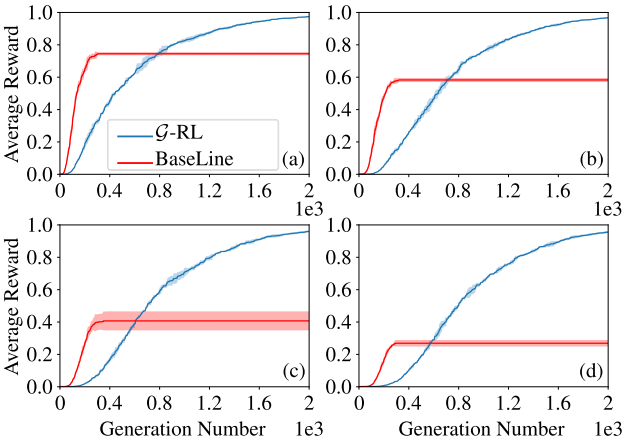


FIGURE 5. Four instances of the variation of the Wahba problem with different sizes (complexity) solved by the CMA-ES algorithm. The size for each is (a) 10, (b) 12, (c) 14, and (d) 16. The mean (solid lines) and the standard deviation (shaded regions) are calculated over five different seeds.

a rotation $\mathbf{R} \in \mathcal{SO}(3)$

$$J(\mathbf{R}) = \frac{1}{2}\sum_{k=1}^{N} a_k \|\mathbf{z}_k - \mathbf{R}\mathbf{y}_k\|^2. \tag{20}$$

where a_k are the weights for each observation. In our case, orientation is represented by unit quaternions. Our experiments use a set of random 3-dimensional vectors and their corresponding rotated ones as the state. The predicted unit quaternion $\hat{\mathbf{Q}}$ is compared to the original rotation $\mathbf{Q}$ with a reward given by $r = -d(\mathbf{Q}, \hat{\mathbf{Q}})$ as in Fig. 3, or by $r = e^{-d(\mathbf{Q},\hat{\mathbf{Q}})}$ as in Fig. 4 and Fig. 5, where $d(\mathbf{Q}_t, \hat{\mathbf{Q}}_t)$ is the distance between two unit quaternions as given by equation (13).

Figure 3 shows the results of learning the orientation represented as a unit quaternion using Gaussian Policy Parameterization (GPP), Geometric Reinforcement Learning ($\mathcal{G}$-RL), and Bingham Policy Parameterization (BPP) [32]. The quality of the learned policy using $\mathcal{G}$-RL was better than GPP for both SAC [8] and PPO [9], while compared to BPP a slightly better policy was learned for SAC and a comparable policy was learned for PPO.

We also used a less complex variation of the Wahba problem by limiting the number of learning orientations to 10, 12, 14, and 16 for PoWER and CMA-ES. As shown in Fig. 4 and Fig. 5, our goal from these experiments is to show the importance of avoiding approximation (normalization) when learning unit quaternions. The results of $\mathcal{G}$-RL are significantly better than the GPP results.

2) SPD MATRIX WAHBA PROBLEM

In addition to quaternions, the Wahba problem was also implemented with SPD matrices manipulating a set of random 3-dimensional vectors. One can think of this problem as a spring system, where the SPD represents the stiffness coefficient of spring and the vector set represent positional displacements. Manipulating the vector set with the SPD provides the spring force at each displacement.

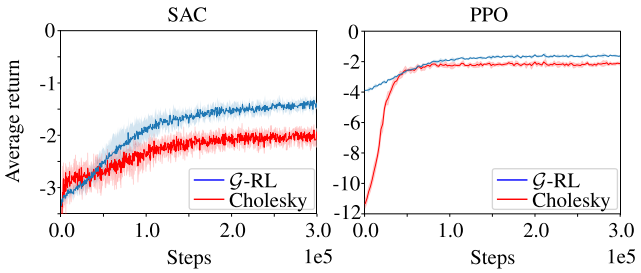


FIGURE 6. Illustrates the quality of learning a policy, utilizing SAC on the left and PPO on the right, in solving a variation of the Wahba problem. The policy's objective is to predict SPD matrices that represent the stiffness coefficient of spring, as well as a set of vectors representing positional displacements. By manipulating the vector set with the SPD matrices, the spring force is determined for each displacement. The resulting curve of our $\mathcal{G}$-RL approach is in blue, while the resulting baseline curve, using Cholesky decomposition, is in red. The solid lines indicate the mean performance across five different random seeds, while the shaded regions represent the standard deviation.

The reward for this version of the problem is given by $r = -d(\mathbf{W}, \widehat{\mathbf{W}})$, where $d(\mathbf{W}, \widehat{\mathbf{W}})$ is the affine invariant distance between the original SPD matrix $\mathbf{W}$ and the predicted SPD matrix $\widehat{\mathbf{W}}$ given by equation (19).

Fig. 6 shows the progression in the quality of learning a policy (specifically, a variation of the Wahba problem) over time. This is depicted using SAC on the left and PPO on the right. The policy in question predicts the SPD that represents the spring's stiffness coefficient, which operates on a vector set that represents positional displacements. The force of the spring at each displacement is derived when the vector set is manipulated with the SPD. In order to ensure a comprehensive comparison, the Cholesky decomposition method, represented in red, is applied as a baseline against our $\mathcal{G}$-RL approach, which is illustrated in blue. Both the mean values (denoted by the solid lines) and the standard

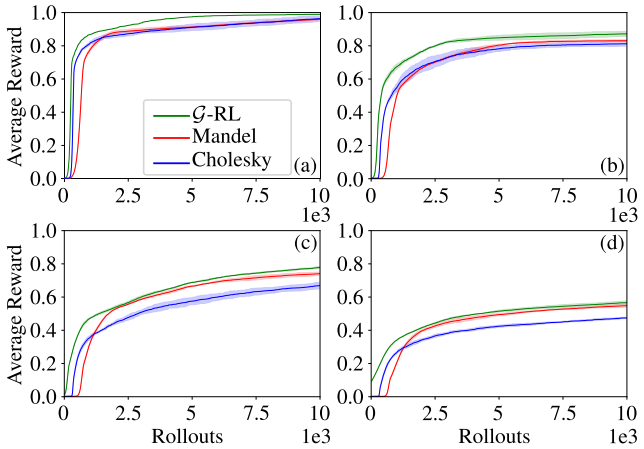


FIGURE 7. The PoWER algorithm is employed to solve four instances of a variation SPD Wahba problem, each with a different size indicating varying levels of complexity. The sizes of the instances are (a) 3, (b) 6, (c) 9, and (d) 12. The solid lines represent the mean performance across five different random seeds, while the shaded regions indicate the standard deviation.

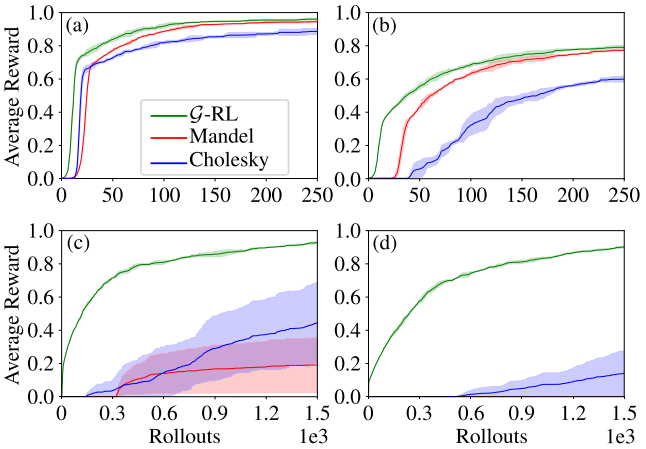


FIGURE 8. The CMA-ES algorithm is employed to solve four instances of a modified SPD Wahba problem, each with a different size indicating varying levels of complexity. The sizes of the instances are as follows: (a) 3, (b) 6, (c) 9, and (d) 12. The solid lines represent the mean performance across five different random seeds, while the shaded regions indicate the standard deviation. Note: the number of rollouts in (c) and (d) is increased to 1500 rollouts to show the significance of the difference between the proposed algorithm and the baseline.

deviation (shown via the shaded regions) are computed over five different seeds. As depicted in the figure, the quality of applying RL algorithms (SAC and PPO) using $\mathcal{G}$-RL is obviously higher than what is achieved using Cholesky.

Testing a simpler version of this problem allows for the opportunity to also evaluate alternative parameterization methods for SPD data, including using Cholesky factorization and Mandel's notation, both of which were evaluated against tangent space parameterization. As seen in Fig. 7 and Fig. 8, PoWER and CMA-ES show that $\mathcal{G}$-RL holds a slight advantage against these other parameterization methods, except for more complex CMA-ES problems where $\mathcal{G}$-RL learns significantly better. Problem sizes 9 and 12 were evaluated under more rollouts to ensure the significance of the comparison.

3) ORIENTATION TRAJECTORY LEARNING PROBLEM

Some manipulation learning problems require learning a desired trajectory of the end-effector pose. In this section, we focus on learning a trajectory of orientations where a policy is trained to follow a well-defined trajectory. The current state (orientation) at time t is an input to the policy, and the policy decides what the next (state) orientation at time $t+1$ should be. The reward captures how close the learned trajectory is to the target one, $r = \sum_{t=1}^{T} e^{-d(\mathbf{Q}_t, \hat{\mathbf{Q}}_t)}$, where $\mathbf{Q}_t$ is the target orientation at time t, $\hat{\mathbf{Q}}_t$ is the predicted orientation at time t, and $d(\mathbf{Q}_t, \hat{\mathbf{Q}}_t)$ is the distance between two unit quaternions as given by equation (13).

Fig. 9 demonstrates the process of learning a policy for regenerating an orientation trajectory for a specific manipulation task, utilizing both PoWER and CMA-ES algorithms. This orientation is denoted by unit quaternions. In the top figure, each unit quaternion is embodied as a 4-dimension vector, each dimension of which records its trajectory through a separate curve. The ultimate policy found using the PoWER algorithm is depicted on the top left. The ground truth is represented with a black dashed line, the normalized baseline is a red solid line, and the $\mathcal{G}$-RL is shown as a blue solid line. Alternatively, the top right portrays the optimal policy identified by the CMA-ES algorithm, displaying the ground truth as a black dashed line, the normalized baseline as a red solid line, and the $\mathcal{G}$-RL as a yellow solid line. In the middle, the figure measures the error, determined through the quaternion distance equation (13), comparing the divergence between the trajectories produced via the RL learned policies and the actual ground-truth. Finally, the figure at the bottom signifies the average reward correlated to the number of rollout trials. We observe that the $\mathcal{G}$-RL error is substantially smaller than the baseline error. When employing the PoWER algorithm, the advantage of using $\mathcal{G}$-RL becomes more apparent. However, in both cases, the algorithms' performance significantly improves by applying $\mathcal{G}$-RL compared to the conventional solution, which is the baseline. This shows that utilizing $\mathcal{G}$-RL on a task that involves a trajectory is advantageous because our suggested algorithm predicts the action within the parameterized tangent space and subsequently parallel transport it to the local tangent space that moves over time. This transition within the tangent spaces assures that we optimally utilize Riemannian geometry. It situates the transported action in close vicinity to the origin of the local tangent space (mapped to the local neighborhood of the origin of the tangent space), thereby providing the most suitable configuration.

4) SPD MATRIX TRAJECTORY LEARNING PROBLEM

As with the Wahba problem, the trajectory learning problem was also replicated using SPD matrices as well, adjusting

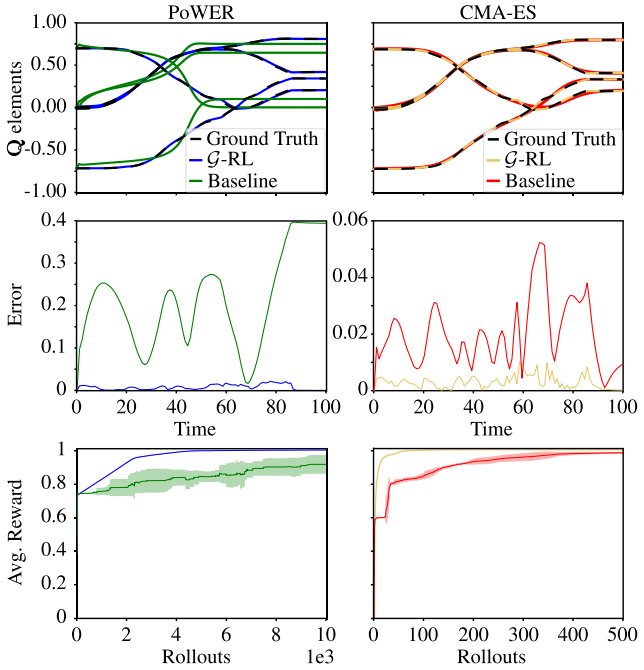


FIGURE 9. Both PoWER (*left*) and CMA-ES (*right*)algorithms are used to learn a policy for orientation trajectory tracking, represented as unit quaternions, in a manipulation task. *Top*: The quaternion tracking response of our $\mathcal{G}$-RL approach is compared with the baseline and ground truth. *Middle*: represents the error, computed using (13), between the trajectories generated by the RL learned policies and the ground truth. *Bottom*: The average reward is plotted with respect to the rollout number, demonstrating the learning progress of the algorithms over time.

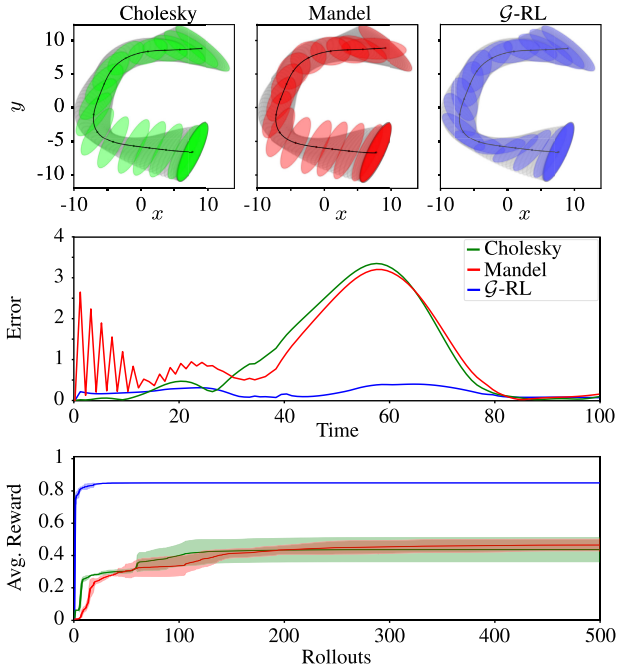


FIGURE 10. Illustrates the learning of a policy using CMA-ES to regenerate the manipulability ellipsoids, adopted from [36]. Three different approaches are compared: two baselines (Cholesky-based and Mandel-based) and our proposed $\mathcal{G}$-RL approach. *Top*: Show the response of tracking a C-shape trajectory in Cartesian space (black dots). Gray ellipsoids represent the ground truth, Cholesky-based ellipsoids are in green, Mandel-based ellipsoids are in red, and our $\mathcal{G}$-RL-based ellipsoids are in blue. *Middle*: Represents the error, computed by (19), between the trajectories generated by the RL learned policies and the ground truth. *Bottom*: Shows the average reward with respect to the rollout number, indicating the learning progress of the algorithms over time.

the policy to learn a trajectory of SPD matrices instead of a trajectory of orientations (quaternions).

The reward for the full trajectory in this problem is given by $r = \sum_{t=1}^{T} e^{-d(\mathbf{W}_t, \widehat{\mathbf{W}}_t)}$ where $\mathbf{W}_t$ is the target SPD matrix at time t, $\widehat{\mathbf{W}}_t$ is the predicted SPD matrix at time t, and $d(\mathbf{W}_t, \widehat{\mathbf{W}}_t)$ is the affine invariant distance between both SPD matrices given by equation (19).

In similar context with the orientation trajectory learning problem, and as depicted in Fig. 10 and 11 both CMA-ES PoWER are used, but the problem here is to regenerate manipulability ellipsoids from [43]. The figures showcases various trajectories: the ground-truth SPD trajectory illustrated with gray ellipsoids, a Cartesian trajectory represented by black dots, two baseline methods - Cholesky-based SPD trajectory illustrated with green ellipsoids (left top), Mandel-based SPD trajectory with red ellipsoids (middle top) - and the proposed $\mathcal{G}$-RL based SPD trajectory shown with blue ellipsoids (right top). Fig. 12 demonstrates the same data with respect to time. Back to Fig. 10 and 11 The middle of the figure showcases the error between the RL learned policies' generated trajectories and the ground-truth values based on the affine invariant distance equation (16). At the bottom portion of the figure, we see an illustration of the average reward in correlation with the rollout number. At first glance at these figures, one can observe that the manipulability ellipsoids generated by $\mathcal{G}$-RL are tracking the ground truth much better than Cholesky and Mandel, and this is quantified by the error figure. Furthermore, the $\mathcal{G}$-RL approach learns faster and converges to a significantly better solution than the commonly used algorithms (Cholesky and Mandel). As we point out about the results of the experiments of the quaternions trajectory learning, applying $\mathcal{G}$-RL on a problem involving a trajectory is most beneficial because our proposed algorithm predicts the action on the parameterization tangent space, then parallel transport it to the local tangent space. This moving tangent space guarantees that we are using the Riemannian geometry in the most appropriate configuration, where the predicted action is located in the close neighborhood of the origin of the local tangent space.

B. REAL EXPERIMENTS (BALL-IN-A-HOLE)

The Ball-in-a-hole problem is a new benchmark proposed in this paper inspired by the Ball-in-a-cup [43] and the ball balancing [44] problems. The problem setup is as depicted in Fig. 13, where a plate with a hole in the middle is attached to the end-effector of the TM5-900 Collaborative Robot (cobot). A camera is also attached to the robot's end effector and is on a stand to always face the surface of the plate. A ping-pong ball is present on the plate, which has its position tracked

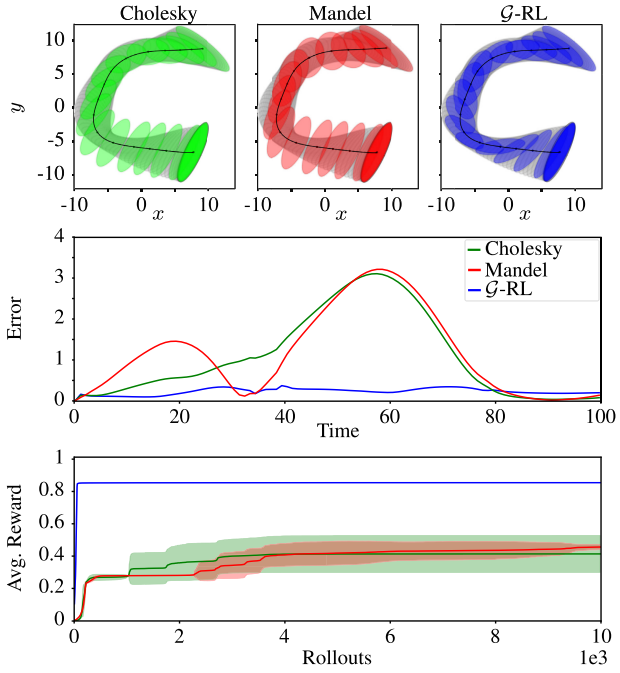


FIGURE 11. Illustrates the learning of a policy using PoWER to regenerate the manipulability ellipsoids, adopted from [36]. Three different approaches are compared: two baselines (Cholesky-based and Mandel-based) and our proposed $\mathcal{G}$-RL approach. *Top*: Show the response of tracking a C-shape trajectory in Cartesian space (black dots). Gray ellipsoids represent the ground truth, Cholesky-based ellipsoids are in green, Mandel-based ellipsoids are in red, and our $\mathcal{G}$-RL-based ellipsoids are in blue. *Middle*: Represents the error, computed by (19), between the trajectories generated by the RL learned policies and the ground truth. *Bottom*: Shows the average reward with respect to the rollout number, indicating the learning progress of the algorithms over time.

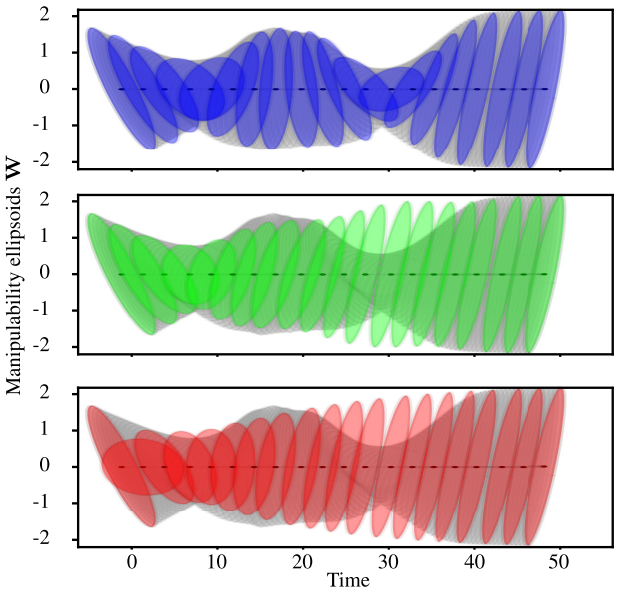


FIGURE 12. Illustrates learned policies quality using CMA-ES to regenerate the manipulability ellipsoids from [36], the ground-truth is depicted as the gray ellipsoids, the proposed $\mathcal{G}$-RL approach is depicted as the blue ellipsoids (top) and two baselines (Cholesky is depicted as the green ellipsoids (middle) and Mandel is depicted as the red ellipsoids (bottom)). The shown trajectories are over time.

by the camera. The robot's end-effector position is fixed, with only its orientation being changed. The state $[\mathfrak{s}_{\mathcal{M}}, \mathfrak{s}_{\mathcal{R}}]$ includes the current orientation of the end-effector (manifold data $\mathfrak{s}_{\mathcal{M}}$) and the current position of the ball on the plate (Euclidean data $\mathfrak{s}_{\mathcal{R}}$). On the parameterization tangent space, we concatenate the Euclidean part with the manifold part and deliver it to the policy. The reward is represented by $\exp^{-d}$, where d is the distance between the center of the ball and the center of the hole measured using the vision system. As this is a challenging problem (lightweight ball, noise in the vision system, and with position control), we decided to start each rollout with the ball in the same initial position.

Regarding the TM5-900 cobot limitations, real-time communication is not guaranteed as all communications pass through the TM-Flow software using a Position, Velocity, Time (PVT) function. No variable impedance control or admittance control is possible as of writing this paper. Therefore, we had to split the trajectory from one rollout into a number of steps. After each orientation change, the ball's location was immediately read and included in the terminal reward (used to guide the RL algorithm). We used the PoWER algorithm to learn a policy that moves the ball into the hole, with Fig. 14 showing the experiment results. The algorithm eventually converged to a local policy, where it learned how to place the ball in the hole via a single axis, as seen in the demonstration video.

VI. DISCUSSION

As noted in the experimental results (Sec. V-A), as the complexity of the problem increased, the advantage of using $\mathcal{G}$-RL over regular approximation approaches is more significant. This allows us to conclude that in moderately complex problems, the error caused by normalization is significant enough to affect the quality of the solution, and there is a clear advantage in using the proposed $\mathcal{G}$-RL.

As already noted in [32], BPP parameterization relies on the prediction from multiple neural networks, which may introduce significant approximation errors. This culminates in an unstable learning process unlike GPP and $\mathcal{G}$-RL. We have experimentally observed this problem of BPP, and several attempts were made before representative results were achieved with this approach. On the contrary, the stability of $\mathcal{G}$-RL was on par with GPP and better than BPP, verifying that the one-to-one mappings between the manifold and tangent space are stable.

We experimentally observed that the average computational overhead of $\mathcal{G}$-RL over GPP is about 3% for SAC and 6% for PPO. These results were expected as the mappings between the tangent space and manifold are not computationally expensive and it is straightforward to implement. This contrasts with BPP, for which we have observed an average overhead of about 33% for SAC and 118% for PPO. Moreover, BPP involves modifying the distribution and customizing the algorithm to fit. Therefore, we conclude

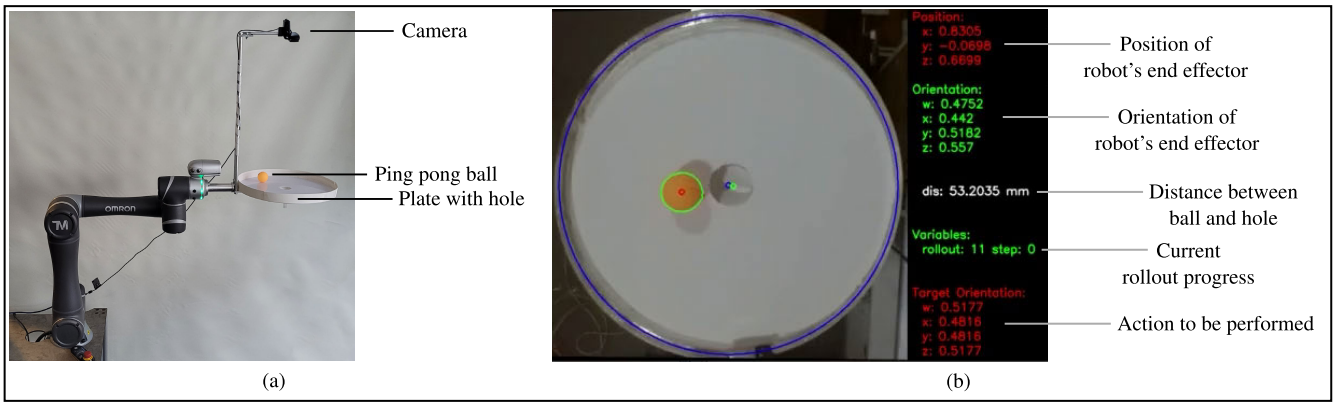


FIGURE 13. (a) Ball-in-a-hole problem setup. A plate with a hole in the middle is attached to the robot's end-effector. The plate's circumference is surrounded by cardboard so the ball does not fall outside the plate. A ping pong ball is located on the top of the plate. A camera is also attached to the end-effector in order to measure the distance between the center of the ball and the center of the hole. (b) shows the plate view using the top camera and the data captured from both the vision system and the robot controller.

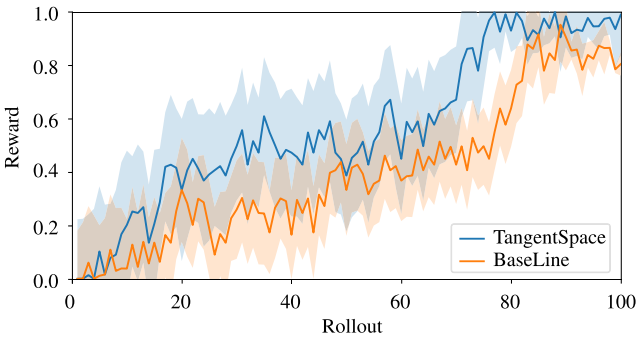


FIGURE 14. The expected return of the learned policy in the Ball-in-a-hole evaluation averaged over five runs.

that $\mathcal{G}$-RL can provide a noticeable improvement in solution quality over GPP at the cost of a small performance penalty and that it can deliver at least equal results to BPP while performing much faster.

Despite the improvement in accuracy, in the case of parameterizing in a single fixed tangent space, where the parameterization and the local tangent spaces to be established at the same fixed point like in the Wahba problem, it would be most beneficial where data points are in the neighborhood of the origin of the tangent space. This is due to the tangent space projection that locally preserves distances near the origin, while distances measured away from the origin are less accurate. Furthermore, this fixed tangent space should be established on or very close to the mean of the data; otherwise, the algorithm's accuracy can be significantly affected.

While this work is limited to the $\mathcal{S}^3$ and $\mathcal{S}^d_{++}$ manifolds, it has the potential to be extended to other non-Euclidean manifolds with the proper investigation and analysis. We leave this as future work.

VII. CONCLUSION

Applying RL algorithms on geometric data like orientation, manipulability, or stiffness is common in robotics, and these algorithms usually perform better when considering the unique structure of these data. The current study was generally dedicated to this topic and showed how RL can be applied to learn geometric actions in the task space (i.e., orientation represented by unit quaternions, and stiffness represented by SPD matrices); parameterization and optimization are carried on the tangent space, and the policy evaluation is carried on the corresponding manifold $\mathcal{M}$. We found that adapting the Gaussian distribution, which is simple and powerful, to the geometry of non-Euclidean data makes it competitive with alternative distributions (e.g., Bingham). Empirical results on both the simulation and the physical robot reflect the importance of considering the geometry of non-Euclidean data and how the performance and accuracy of the overall learning process are consequently affected. This research holds promising potential for broader applications in the future. The methodology can be adapted to extend to various other manifolds. Moreover, its applicability in specialized areas such as industrial manipulation tasks, e.g., polishing applications, further highlights the versatility of this work.

As a future work, the focus will shift to augmenting this framework in the context of model-based RL algorithms. This requires a deeper investigation and the development of a novel approach to efficiently represent the stochastic transition dynamics on different manifolds.

ACKNOWLEDGMENT

The authors would like to thank Ville Kyrki of Aalto University and Luis Figueredo of MIRMI, Technical University of Munich for their help and support in reading and reviewing the math flow of the proposed approach.

REFERENCES

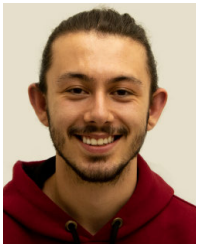
[1] F. J. Abu-Dakka, Y. Huang, J. Silvério, and V. Kyrki, "A probabilistic framework for learning geometry-based robot manipulation skills," *Robot. Auto. Syst.*, vol. 141, Jul. 2021, Art. no. 103761.

[2] X. Pennec, "Intrinsic statistics on Riemannian manifolds: Basic tools for geometric measurements," *J. Math. Imag. Vis.*, vol. 25, no. 1, pp. 127–154, Jul. 2006.

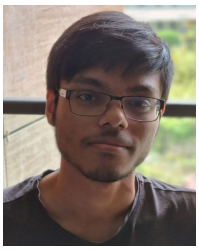
[3] S. Calinon, "Gaussians on Riemannian manifolds: Applications for robot learning and adaptive control," *IEEE Robot. Autom. Mag.*, vol. 27, no. 2, pp. 33–45, Jun. 2020.
[4] M. Chen, A. Liu, W. Liu, K. Ota, M. Dong, and N. N. Xiong, "RDRL: A recurrent deep reinforcement learning scheme for dynamic spectrum access in reconfigurable wireless networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 2, pp. 364–376, Mar. 2022.
[5] M. Chen, W. Liu, T. Wang, S. Zhang, and A. Liu, "A game-based deep reinforcement learning approach for energy-efficient computation in MEC systems," *Knowl.-Based Syst.*, vol. 235, Jan. 2022, Art. no. 107660.
[6] Y. Ren, W. Liu, A. Liu, T. Wang, and A. Li, "A privacy-protected intelligent crowdsourcing application of IoT based on the reinforcement learning," *Future Gener. Comput. Syst.*, vol. 127, pp. 56–69, Feb. 2022.
[7] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.
[8] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.
[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[10] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation*. Berlin, Germany: Springer, 2006, pp. 75–102.
[11] F. Stulp and O. Sigaud, "Policy improvement: Between black-box optimization and episodic reinforcement learning," in *Journées Francophones Planification, Décision, et Apprentissage pour la conduite de systèmes*, 2013.
[12] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, Nov. 2010.
[13] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Mach. Learn.*, vol. 84, nos. 1–2, pp. 171–203, Jul. 2011.
[14] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Jul. 2017, pp. 51–58.
[15] M. Saveriano, Y. Yin, P. Falco, and D. Lee, "Data-efficient control policy search using residual dynamics learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4709–4715.
[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[17] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[18] Y. Huang, F. J. Abu-Dakka, J. Silvério, and D. G. Caldwell, "Toward orientation learning and adaptation in Cartesian space," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 82–98, Feb. 2021.
[19] H. Beik-Mohammadi, S. Hauberg, G. Arvanitidis, G. Neumann, and L. Rozo, "Learning Riemannian manifolds for geodesic motion skills," 2021, *arXiv:2106.04315*.
[20] C. Chang, K. Haninger, Y. Shi, C. Yuan, Z. Chen, and J. Zhang, "Impedance adaptation by reinforcement learning with contact dynamic movement primitives," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2022, pp. 1185–1191.
[21] I. Kao, M. R. Cutkosky, and R. S. Johansson, "Robotic stiffness control and calibration as applied to human grasping tasks," *IEEE Trans. Robot. Autom.*, vol. 13, no. 4, pp. 557–566, 1997.
[22] X. Zhang, L. Sun, Z. Kuang, and M. Tomizuka, "Learning variable impedance control via inverse reinforcement learning for force-related tasks," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2225–2232, Apr. 2021.
[23] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Force-based learning of variable impedance skills for robotic manipulation," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots (Humanoids)*, Jul. 2018, pp. 1–9.
[24] M. Oikawa, K. Kutsuzawa, S. Sakaino, and T. Tsuji, "Assembly robots with optimized control stiffness through reinforcement learning," 2020, *arXiv:2002.12207*.
[25] S. Colutto, F. Fruhauf, M. Fuchs, and O. Scherzer, "The CMA-ES on Riemannian manifolds to reconstruct shapes in 3-D voxel images," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 227–245, Apr. 2010.
[26] J. Chen, Y. Yin, T. Birdal, B. Chen, L. J. Guibas, and H. Wang, "Projective manifold gradient layer for deep rotation regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Aug. 2022, pp. 6646–6655.
[27] L. Rozo and V. Dave, "Orientation probabilistic movement primitives on Riemannian manifolds," in *Proc. Conf. Robot Learn.*, 2022, pp. 373–383.
[28] D. Wang, R. Walters, and R. Platt, "So (2) equivariant reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022, pp. 1–19.
[29] N. Jaquier, L. Rozo, S. Calinon, and M. Bürger, "Bayesian optimization meets Riemannian manifolds in robot learning," in *Proc. Conf. Robot Learn.*, 2020, pp. 233–246.
[30] N. Jaquier, V. Borovitskiy, A. Smolensky, A. Terenin, T. Asfour, and L. Rozo, "Geometry-aware Bayesian optimization in robotics using Riemannian matérn kernels," in *Proc. Conf. Robot Learn.*, 2022, pp. 794–805.
[31] D. Wang, C. Kohler, and R. Platt, "Policy learning in SE(3) action spaces," 2020, *arXiv:2010.02798*.
[32] S. James and P. Abbeel, "Bingham policy parameterization for 3D rotations in reinforcement learning," 2022, *arXiv:2202.03957*.
[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[34] J. Jost and J. Jost, *Riemannian Geometry and Geometric Analysis*, vol. 42005. Cham, Switzerland: Springer, 2008.
[35] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[36] F. J. Abu-Dakka and V. Kyrki, "Geometry-aware dynamic movement primitives," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4421–4426.
[37] M. J. Zeestraten, "Programming by demonstration on Riemannian manifolds," Ph.D. dissertation, Dept. Inform., Bioeng., Robot., Syst. Eng., Dept. Adv. Robot. Istituto Italiano di Tecnologia, Univ. Genova, Genoa, Italy, 2018.
[38] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC Press, 2017.
[39] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2007.
[40] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, Jan. 2006.
[41] S. Sra and R. Hosseini, "Conic geometric optimization on the manifold of positive definite matrices," *SIAM J. Optim.*, vol. 25, no. 1, pp. 713–739, Jan. 2015.
[42] G. Wahba, "A least squares estimate of satellite attitude," *SIAM Rev.*, vol. 7, no. 3, p. 409, Jul. 1965.
[43] C. Sumners, *Toys in Space: Exploring Science With the Astronauts*. IOP Publishing, 1994. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/638/1/012004
[44] M. M. Kopichev, A. V. Putov, and A. N. Pashenko, "Ball on the plate balancing control system," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 638, no. 1, 2019, Art. no. 012004.

NASEEM ALHOUSANI received the B.Sc. degree in computer science and the M.Sc. degree in scientific computing from Birzeit University, in 2003 and 2006, respectively. He is currently pursuing the Ph.D. degree in computer engineering with Istanbul Technical University, İstanbul, Turkey. From 2006 to 2015, he was a Lecturer with the Computer Science Department, Palestine Technical University-Kadoorie. Since 2015, he has been a Researcher with ILITRON Energy and Technology, İstanbul. His research interests include reinforcement learning, planning, and learning on Riemannian manifolds.

MATTEO SAVERIANO (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in automatic control engineering from the University of Naples, Italy, in 2008 and 2011, respectively, and the Ph.D. degree from the Technical University of Munich, in 2017. Currently, he is an Assistant Professor with the Department of Industrial Engineering (DII), University of Trento, Italy. Previously, he was an Assistant Professor with the University of Innsbruck and a Postdoctoral Researcher with the German Aerospace Center (DLR). He is an Associate Editor of IEEE Robotics and Automation Letters and *IJRR*. His research interests include robot learning, human–robot interaction, understanding, and interpreting human activities. For more information visit the link: https://matteosaveriano.weebly.com/.



IBRAHIM SEVINC received the B.Sc. degree in electronics and communication engineering from Istanbul Technical University, in 2023. He has been with MCFLY Robot Technologies, İstanbul, Turkey, since 2022.



TALHA ABDULKUDDUS is currently pursuing the B.Sc. degree in computer science with King's College London, U.K. Since 2022, he has been with ILITRON Energy and Information Technologies, İstanbul, Turkey.

HATICE KOSE (Member, IEEE) received the Ph.D. degree from the Computer Engineering Department, Boğaziçi University, Turkey. She has been a Full Professor with the Faculty of Computer and Informatics Engineering, Istanbul Technical University, Turkey, coordinating the GameLaboratory and Cognitive Social Robotics Laboratory, since 2010. From 2006 to 2010, she was a Research Fellow with the University of Hertfordshire. Her current research interests include gesture communication (involving sign language) and imitation-based interaction games with social humanoid robots for the education and rehabilitation of children with hearing impairment and children with ASD. She is leading several national projects and taking part in several Horiizon2020 projects, Erasmus+ and Cost actions, on social assistive robots, sign language tutoring robots, and human–robot interaction.

FARES J. ABU-DAKKA (Member, IEEE) received the B.Sc. degree in mechanical engineering from Birzeit University, Palestine, in 2003, and the D.E.A. and Ph.D. degrees in robotics motion planning from the Polytechnic University of Valencia, Spain, in 2006 and 2011, respectively.

In 2012, he started his first postdoctoral with the Jozef Stefan Institute, Slovenia. Between 2013 and 2016, he held a Visiting Professorship with ISA, Carlos III University of Madrid, Spain. From 2016 to 2019, he was a Postdoctoral Researcher with Istituto Italiano di Tecnologia (IIT). From 2019 to 2022, he was a Research Fellow with Aalto University. Then, in 2022, he moved to MIRMI, Technical University of Munich, Germany, to serve as a Senior Scientist and a Leader of the Robot Learning Group. His research interests include the intersection of control theory, differential geometry, and machine learning, in order to enhance robot manipulation performance and safety.

Dr. Abu-Dakka served as an Associate Editor for *ICRA*, *IROS*, and IEEE Robotics and Automation Letters. For more information visit the link: https://sites.google.com/view/abudakka/.