Reinforcement Learning with Lie Group Orientations for Robotics

Martin Schuck, Jan Brüdigam, Sandra Hirche, and Angela Schoellig

Abstract-Handling orientations of robots and objects is a crucial aspect of many applications. Yet, ever so often, there is a lack of mathematical correctness when dealing with orientations, especially in learning pipelines involving, for example, artificial neural networks. In this paper, we investigate reinforcement learning with orientations and propose a simple modification of the network's input and output that adheres to the Lie group structure of orientations. As a result, we obtain an easy and efficient implementation that is directly usable with existing learning libraries and achieves significantly better performance than other common orientation representations. We briefly introduce Lie theory specifically for orientations in robotics to motivate and outline our approach. Subsequently, a thorough empirical evaluation of different combinations of orientation representations for states and actions demonstrates the superior performance of our proposed approach in different scenarios, including: direct orientation control, end effector orientation control, and pick-and-place tasks.

I. INTRODUCTION

The orientation of robots, end effectors, or objects is a central state in almost all robotics applications, such as drone flying [1], [2], [3], highly dynamic locomotion [4], [5], or manipulation of objects [6], [7], [8]. While positions can be trivially described with Cartesian coordinates in Euclidean space, orientations require more attention. Since orientations form a mathematical group, they have favorable structural properties that can simplify and improve calculations when respected [9], [10]. However, in practice, this group structure is often only partially considered or even entirely ignored, leading to, for example, issues with singularities in Euler angles. Moreover, learning orientations with artificial neural networks often conflicts with the group structure because such networks typically operate in Euclidean space, i.e., \mathbb{R}^n , since most practical learning implementations provide full support only for such a representation [11], [12]. Another complexity is that orientations can be expressed by many different representations, which have different advantages and disadvantages, such as computational speed, smoothness, multi-cover, and singularities [13].

This paper focuses on treating orientations mathematically consistently in reinforcement learning (RL). Our central claim is that in an RL setting, adhering to the Lie group structure of orientations where possible results in mathematically sound expressions and practically superior performance in learning progress, computational speed, and policy performance. Specifically, our contributions are:

 A modification of network inputs and outputs in reinforcement learning based on the Lie algebra of orientations that is mathematically sound, practically efficient

 $\xrightarrow{s \in \mathcal{M}} \operatorname{Log}^{\mathcal{E}} \xrightarrow{\tau_s \in \mathbb{R}^3} \xrightarrow{s_{\tau_a} \in \mathbb{R}^3} \operatorname{Exp}^{a \in \mathcal{M}} \xrightarrow{s \cdot a} \xrightarrow{s' \in \mathcal{M}}$



Fig. 1. **Top**: Our proposed learning architecture. Starting with an orientation state $s \in \mathcal{M}$, we take the Log to obtain a vector ${}^{\mathcal{E}}\boldsymbol{\tau}_s \in \mathbb{R}^3$ in the tangent space. This vector is passed into the neural network to obtain an action vector ${}^{s}\boldsymbol{\tau}_a \in \mathbb{R}^3$ relative to s. By taking the Exp, we obtain a relative action $a \in \mathcal{M}$ which is composed with the original state s to obtain the new state $s' = s \cdot a \in \mathcal{M}$. **Bottom**: The hardware setup for the pick-and-place task. A cube is moved from an initial pose to a goal pose in the air.

to implement, and leads to improved policy performance compared to other common implementations.

- A practical introduction to Lie theory for orientations in robotics to motivate and outline the modified network architecture.
- An empirical evaluation of different combinations of orientation representations for observations and actions to determine the most suitable representation for RL with orientations.

A. Problem setting

We consider reinforcement learning (RL) for a goalconditioned Markov decision process with state $s \in S$, goal $g \in \mathcal{G}_s$, action $a \in \mathcal{A}$, and sparse reward $r \in \{-1, 0\}$. The state can be composed of positions and orientations. The actions are position or orientation commands relative to the current state, resulting in the dynamics s' = f(s, a).

The objective in goal-conditioned RL is to find a policy

All authors are with Technical University of Munich

 $oldsymbol{a}=\pi^{\star}(oldsymbol{s},oldsymbol{g})$ that maximizes the expected future reward:

$$\boldsymbol{\pi}^{\star} = \operatorname*{arg\,max}_{\boldsymbol{\pi}} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^{t} r(\boldsymbol{s}_{t}, \boldsymbol{g}_{t}, \boldsymbol{\pi}(\boldsymbol{s}_{t}, \boldsymbol{g}_{t})) \right], \quad (1)$$

with discount factor $\gamma \in [0, 1)$.

For the reward, we consider the distance between the state and the goal. For Euclidean quantities, i.e., positions, the distance is the norm of the state-to-goal difference:

$$d_{\mathrm{E}}(\boldsymbol{s}, \boldsymbol{g}) = \|\boldsymbol{s} - \boldsymbol{g}\| \in \mathbb{R}.$$
 (2)

For orientations, the distance is the norm of the orientation difference in the tangent space between state and goal orientations:

$$d_{\mathcal{M}}(\boldsymbol{s},\boldsymbol{g}) = \|\boldsymbol{s} \ominus \boldsymbol{g}\| \in \mathbb{R}.$$
 (3)

This distance is simply the angle between the two orientations, independent of the representation, and, therefore, comparable between different representations. We explain how this difference is computed in Sec. II-B and refer to [10] for further details and notation.

The sparse reward is 0 if the Euclidean distance is smaller than a threshold $\epsilon_{\rm E} \in \mathbb{R}^+$ and the orientation distance is smaller than a threshold $\epsilon_{\mathcal{M}} \in \mathbb{R}^+$, and -1 otherwise, i.e.,

$$r(\boldsymbol{s}, \boldsymbol{g}, \boldsymbol{a}) = \begin{cases} 0 & \text{if } d_{\rm E} \leq \epsilon_{\rm E} \text{ and } d_{\mathcal{M}} \leq \epsilon_{\mathcal{M}} \\ -1 & \text{otherwise} \end{cases} .$$
(4)

For our experiments, we use the deep deterministic policy gradient (DDPG) learning algorithm [14] with hindsight experience replay [15].

B. Related work

The correct treatment of orientations in robotics is gaining increasing interest to improve performance. Learning methods based on artificial neural networks require careful attention due to their Euclidean structure in practice, which is not directly compatible with the manifold structure of orientations. Accordingly, [16] investigates which orientation representations exhibit discontinuous maps to the rotation group SO(3) and how these discontinuities lead to theoretical and practical learning deficiencies. Continuing in this direction, [17] investigates supervised learning with orientations both at the input and output of networks. While both works address differences in rotation representations for supervised learning, they still treat the representations as Euclidean vectors, requiring, for example, the projection of the network output onto the orientation manifold. In contrast, we propose a method that adheres to the orientation structure both at the network input and output.

One approach to treat the orientation manifold structure correctly in reinforcement learning can be found in [18]. Here, the focus is on orientations in the action space, which are treated as a Riemannian manifold at the network output. This view can be complex and, if not handled correctly, is subject to subtle issues described in detail in [19]. Initial ideas in [20], where instead of the Riemannian manifold view, the Lie group perspective is taken, simplify computations for orientations. However, the real-world experiments in [20] are limited, and no study of the mutual influence of representation combinations in the action and observation space is considered. In our paper, we conduct a thorough study across many different parametrizations and show results for complex, real-world robotic tasks.

A fundamentally different approach is to adapt the entire network architecture to adhere to the group structure of orientations. Several authors have investigated such methods [21], [22], [23], [24], and there are a number of works specifically on quaternion representations [25], [26]. However, since most state-of-the-art learning libraries have limited support for such network structures, we focus on modifying the network input and output to leverage existing packages and provide a practically beneficial method.

II. BACKGROUND ON LIE THEORY FOR ORIENTATIONS

Unlike the position of an object, its orientation is a non-Euclidean state. An intuitive example of this fact is that incrementally rotating an object will decrease the global orientation once the orientation exceeds π . Nonetheless, the set of all possible orientations can be associated with a mathematical group on a smooth manifold, which has the same shape everywhere, making it a Lie group. In this section, we often follow the excellent introduction to Lie groups for robotics by [10] and refer to this work for further details. In our paper, we only provide the background necessary for learning in robotics with orientations.

First, we provide representations that are commonly used to describe orientations. Then, we explain how distances between orientations can be computed and how relative orientations can be composed. Finally, we give insights into how these representations are used in learning with neural networks.

A. Lie groups as orientation representations

The orientation of an object is a geometric property. In order to work with orientations, we need a mathematical representation. Since there are different applications with distinct requirements, various orientation representations are commonly used, and a summary of common representations is given in Tab. I.

Representation	Structure	Size	Dim.	Cover
Rotation matrices	Lie Group $SO(3)$	9	3	single
3×2 matrices	Lie Group $SO_{1:2}(3)$	6	3	single
Quaternions	Lie Group S^3	4	3	double
Quaternions+	Group	4	3	single
Euler angles	Vector space \mathbb{R}^3	3	3	multi
Axis-angle	Lie Algebra $\mathfrak{m} \cong \mathbb{R}^3$	3	3	multi
	TABLE I			

ORIENTATION REPRESENTATIONS

Some of these representations are a group. Citing [10], a group (\mathcal{G}, \cdot) is a set, \mathcal{A} , with a composition operation, \cdot , that,

for elements $x, y, z \in \mathcal{G}$, satisfies the following axioms:

Closure under
$$\cdot$$
: $\boldsymbol{x} \cdot \boldsymbol{y} \in \mathcal{G}$ (5a)

Identity
$$\mathcal{E}$$
: $\mathcal{E} \cdot x = x \cdot \mathcal{E} = x$ (5b)

Inverse
$$x^{-1}$$
: $x^{-1} \cdot x = x \cdot x^{-1} = \mathcal{E}$ (5c)

Associativity:
$$(\boldsymbol{x} \cdot \boldsymbol{y}) \cdot \boldsymbol{z} = \boldsymbol{x} \cdot (\boldsymbol{y} \cdot \boldsymbol{z})$$
 (5d)

As mentioned above, if the elements of a group exist on a smooth manifold that looks the same everywhere, it is a Lie group. An example is the group of rotation matrices SO(3) with the composition being matrix-matrix multiplication, the identity element being the identity matrix, $\mathcal{E} = I_{3\times 3}$, and the inverse defined as the matrix transpose, $R^{-1} = R^{\mathsf{T}}$.

Since the 3×3 rotation matrices $\mathbf{R} \in SO(3)$ evolve smoothly on their manifold, they form a Lie group. There is a bijective (one-to-one) mapping between the elements of the SO(3) group and the elements of the set of all possible orientations. Therefore, SO(3) is typically referred to as *the* orientation (or rotation) group.

A related and less common yet practically useful representation are the 3×2 matrices $\mathbf{R} \in SO_{1:2}(3)$ composed of the first two columns of the matrices in SO(3) [16]. Given a matrix in $SO_{1:2}(3)$, the third column can be uniquely reconstructed from the first two with the cross product, making $SO_{1:2}(3)$ isomorphic (same structure but different representation) to SO(3), i.e., $SO_{1:2}(3) \cong SO(3)$. The advantage of fewer parameters in the $SO_{1:2}(3)$ representation comes at the cost of losing direct matrix-matrix or matrixvector multiplication.

Another way to represent orientations is with the computationally efficient four-parameter unit quaternions $q \in S^3$, with quaternions $q \in \mathbb{H}$, ||q|| = 1. Unit quaternions also form a Lie group, but there is a surjective (here manyto-one) mapping from S^3 to SO(3). Specifically, there is a double cover, i.e., both q and -q represent the same orientation. Yet, the group of unit quaternions, as well as unit-quaternion distances and incremental changes, can be formulated smoothly.

There are also orientation representations that are not Lie groups. In order to avoid non-uniqueness issues with the double cover, unit quaternions can be limited to a single cover S^{3+} by negating quaternions with negative real part. While S^{3+} still satisfies the group axioms (5) by modifying the composition operation to always negate negative-real-part quaternions, this modification breaks the Lie group structure because of the loss of a smooth manifold.

Another widespread representation is three-parameter Euler angles $e \in \mathbb{R}^3$. Euler angles have a non-smooth mapping to orientations at specific configurations, i.e., singularities. In such configurations, a small local change of orientation can only be achieved by a discontinuity in Euler angles, as shown in Fig. 2. Euler angles also have a surjective mapping to orientations, where in singularities, infinitely many Euler angles represent the same orientation.

Yet another representation is axis-angle $\theta \in \mathbb{R}^3$. This representation builds on the fact that any representation can be expressed as a three-dimensional unit-norm rotation axis



Fig. 2. A comparison of continuous and discontinuous orientation representations. **Top**: the initial frame (1) is first turned -90° around the local *y*-axis to obtain frame (2), and then turned 90° around the local *z*-axis to obtain frame (3). **Bottom**: Representing this frame transformation with a rotation matrix or a quaternion (with double cover) evolves continuously on their respective manifolds. Using Euler angles results in a discontinuity at the singularity.

 $\boldsymbol{u} \in \mathbb{R}^3$ and a rotation angle $\boldsymbol{\theta} \in \mathbb{R}$, resulting in $\boldsymbol{\theta} = \boldsymbol{\theta} \boldsymbol{u}$. There is a surjective mapping from axis-angle to orientations with a multi-cover. A different perspective is to consider this representation as an isomorphism to the Lie algebras for the Lie groups SO(3), $SO_{1:2}(3)$, and S^3 , which we discuss in the next section.

B. Lie algebra for distances and increments

Every Lie group has an associated Lie algebra closely related to the group. The Lie algebra is a vector space tangent to the group at the identity element, and its elements can be represented by vectors. That is, the Lie algebra m is the tangent space $T_{\mathcal{E}}\mathcal{M}$ at the identity \mathcal{E} of the Lie group \mathcal{M} . The relationship between Lie group and algebra is visualized in Fig. 3. Elements $\tau^{\wedge} \in \mathfrak{m}$ of the Lie algebra m can be uniquely represented by vectors $\tau \in \mathbb{R}^m$, where m is the dimension of the group, i.e., $\mathfrak{m} \cong \mathbb{R}^m$. Table II contains orientation Lie groups, Lie algebras, and the Lie algebra vector representations.

Lie Group \mathcal{M}	Lie Algebra m	Vector Representation \mathbb{R}^m	
SO(3)	$\boldsymbol{\theta}^{\times} \in \mathfrak{so}(3)$	$oldsymbol{ heta} \in \mathbb{R}^3$	
$SO_{1:2}(3)$	$\cong \boldsymbol{\theta}^{\times} \in \mathfrak{so}(3)$	$oldsymbol{ heta}\in\mathbb{R}^3$	
50	$\theta/2 \in \mathbb{H}_p$	$oldsymbol{ heta} \in \mathbb{R}^{3}$	
	TABLE	II	

The \times operation in θ^{\times} creates a 3×3 skew-symmetric matrix from θ . For orientation Lie groups such as SO(3), $SO_{1:2}(3)$, and S^3 , the elements of the Lie algebra represented as vectors are axis-angles $\theta \in \mathbb{R}^3$.

There exists a simple relation between a Lie group element $x \in \mathcal{M}$ and its associated Lie algebra element $\tau^{\wedge} \in \mathfrak{m}$ (and



Fig. 3. Visualization of a Lie group on manifold \mathcal{M} , its Lie algebra \mathfrak{m} , i.e., the tangent space at the identity $T_{\boldsymbol{\varepsilon}}\mathcal{M}$, and a tangent space at $T_{\boldsymbol{x}}\mathcal{M}$ at $\boldsymbol{x} \in \mathcal{M}$, where the following holds: $\boldsymbol{y} = \boldsymbol{x} \cdot \operatorname{Exp}(^{\boldsymbol{x}}\boldsymbol{\tau})$.

 $oldsymbol{ au} \in \mathbb{R}^m$):

$$\boldsymbol{x} = \exp(\boldsymbol{\tau}^{\wedge}) = \operatorname{Exp}(\boldsymbol{\tau}),$$
 (6a)

$$\boldsymbol{\tau}^{\wedge} = \log(\boldsymbol{x}), \tag{6b}$$

$$\boldsymbol{\tau} = \operatorname{Log}(\boldsymbol{x}). \tag{6c}$$

The Lie algebra allows us to easily compose relative orientations and compute orientation differences. An incremental orientation ${}^{x}\tau \in T_{x}\mathcal{M}$ expressed in the local tangent space at the orientation $x \in \mathcal{M}$ is achieved by the following composition:

$$\boldsymbol{y} = \boldsymbol{x} \oplus^{\boldsymbol{x}} \boldsymbol{\tau} = \boldsymbol{x} \cdot \operatorname{Exp}(^{\boldsymbol{x}} \boldsymbol{\tau}). \tag{7}$$

Because of the group composition and exponential map, adding a relative orientation change remains on the group manifold and does not require any projection. Similarly, the difference between two orientations can be computed as

$${}^{\boldsymbol{x}}\boldsymbol{\tau} = \boldsymbol{y} \ominus \boldsymbol{x} = \operatorname{Log}(\boldsymbol{x}^{-1}\boldsymbol{y}). \tag{8}$$

With the group operations (5) and the Lie algebra, we can also define a distance metric $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ with the following properties for elements $x, y, z \in \mathcal{M}$:

Zero self distance : $d(\boldsymbol{x}, \boldsymbol{x}) = 0$ (9a)

Positive for
$$x \neq y$$
: $d(x, y) > 0$ (9b)

Symmetry:
$$d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x})$$
 (9c)

Friangle inequality :
$$d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z})$$
. (9d)

Defining such a metric is not trivial: if we want to know the distance of an orientation to the origin, i.e., the identity, we cannot simply take the norm of the orientation representation element as we would do for Euclidean vectors since, for example, any rotation matrix, including the identity matrix, has a norm $||\mathbf{R}||_{\rm F} = \sqrt{3}$. Accordingly, several different metrics exist that measure the distance between orientations [27]. However, we consider the geodesic distance to be the geometrically "correct" metric. The geodesic distance is simply the rotation angle between two orientations.

The Lie algebra is an axis-angle representation and can be used directly as a distance measure. As a result, the distance to the origin for an orientation $x \in \mathcal{M}$ is the norm of its Lie algebra element: $\|\log(x)\| = \|\theta\| = \theta$. The distance between

two orientations $x, y \in \mathcal{M}$ is the distance of the relative orientation between the two to the origin: $\|\log(y^{-1}x)\|$.

Since Euler angles do not constitute a Lie group, we treat them as a vector space and compose orientations by adding Euler angles. Distances are computed by first converting Euler angles to a Lie group representation and then computing the Lie algebra distance.

III. LEARNING WITH ORIENTATIONS

Neural networks can be interpreted as learning a nonlinear feature representation for an input and producing a weighted output of these features. Most network architectures operate in Euclidean vector spaces, i.e., they have vector inputs and outputs, as well as addition and scalar multiplication as operations. Accordingly, the feature representation for orientations happens in Euclidean space by representing orientation representations as Euclidean vectors at the inputs and outputs.

When using Lie group representations and directly passing them into a network, they are mathematically incorrectly treated as vectors at the input, for example, by flattening a rotation matrix $\mathbf{R}_{in} \in SO(3)$ into a vector $\mathbf{r}_{in} \in \mathbb{R}^9$. More problematically, the network's output is also a vector and not on the orientation manifold SO(3). Therefore, some form of projection back onto the manifold is required, for example, based on singular-value decomposition, to obtain the "closest" rotation matrix $\mathbf{R}_{out} \in SO(3)$ from a non-SO(3) matrix $\mathbf{R}'_{out} \in \mathbb{R}^{3\times 3}$.

As an alternative, we propose to properly consider the Lie group structure of orientations when learning with neural networks. To this end, instead of directly feeding an orientation state s as a vector at the input, we pass in the associated Lie algebra element ${}^{\mathcal{E}}\tau_s = \text{Log}(s) \in \mathbb{R}^3$ which is, in fact, a vector. The network then performs operations that are within the algebraic structure of the Lie algebra and also provides an output that is a Lie algebra element ${}^{s}\tau_a \in \mathbb{R}^3$. An orientation action is recovered from this element by taking the exponential $a = \text{Exp}({}^{s}\tau_a) \in \mathcal{M}$. Finally, the new state s' is obtained by composing the current state s and action $a: s' = s \cdot a \in \mathcal{M}$. The full process is visualized in Fig. 1.

In practice, the exponential and logarithm can either be manually implemented in closed form (cf. [10]) or by using the matrix and quaternion exponential and logarithm existing in most linear algebra packages.

IV. EXPERIMENTS

We investigate three settings with increasing complexity to provide empirical results on reinforcement learning with orientations:

- A. Directly learning a policy for orienting a frame with incremental rotation actions without any robot embodiment to obtain a "clean" baseline.
- B. Moving the end effector of a robot arm to a desired orientation to investigate the learning of embodied orientation.



Fig. 4. Comparison of different orientation representations for state and action: Lie algebra m, rotation matrices SO(3) (SO_3), two-column rotation matrices $SO_{1:2}(3)$ ($SO_3^{1:2}$), positive-real-part quaternions \mathcal{S}^{3+} , quaternions \mathcal{S}^3 , Euler angles \measuredangle , and Riemannian manifold action RM. **Top row**: Results for direct orientation control. **Bottom row**: Results for end effector orientation control. **Left column**: Average success rate during training to measure convergence speed and overall success. Higher (blue) is better. **Center column**: Final success rate to measure best task success. Higher (blue) is better. **Right column**: Average reward per step of the final policy to measure policy performance. Closer to zero (blue) is better.

C. Picking and placing a cube with the robot arm where the orientation of the unactuated cube cannot be controlled directly.

We use MuJoCo [28] for the robot simulation, and DDPG for learning, although we repeated some of the experiments with TD3 for verification and obtained similar results.

A. Direct orientation control

We compare 36 orientation representation combinations by evaluating the six representations listed in Tab. I for the state and action. The task is to rotate an initial frame into a goal frame by applying incremental rotation actions, visualized in



Fig. 5. Task progression for direct orientation control. From the initial state s_0 , relative rotation actions a_i are taken to move toward the goal (not shown).

Fig. 5. The initial state s and goal g are sampled uniformly in S^3 and then converted to other representations. For all representations, the action is limited to a relative rotation angle of at most 0.1π . We also repeated the experiments with 0.05π and 0.2π and obtained similar results. During each episode, the agent can take 50 steps to reach the goal. We train the agent for 200,000 environment steps.

We also compare to the Riemannian manifold approach in [18]. Note that the method in [18] only provides a different way of treating actions, but not states.

The results are shown on the top row of Fig. 4. We show three metrics: The average success rate during the training process indicates how quickly and consistently training progress is made, and higher values represent faster training progress. The final success rate at the end of training indicates if the policy is at all able to solve the task. And the average reward per step during 160 rollouts of the final policy shows how well the policy performs, as reaching the goal faster yields a higher reward. For all results, we take the average of 100 runs.

A comparison of the required time for a full training run

is provided in Tab. III. We use the same representation for state and action and show the time spent on network training and policy rollout.

Time (s)	m	SO_3	$SO_{3}^{1:2}$	\mathcal{S}^{3+}	\mathcal{S}^3	4	RM
Train Rollout	18.8 19.5	29.0 38.6	20.5 24.8	27.0 21.3	25.9 19.9	45.1 20.0	28.9 32.7
			TABLE	ш			

TIME SPENT DURING TRAINING (SEE FIG. 4 FOR LABELS)

B. End effector orientation control

We evaluate the applicability of the baseline results to robotic systems by requiring the end effector of a robot arm to reach a desired goal frame from the single initial robot configuration. We compare the same 36 orientation representation combinations for state and action as in Sec. IV-A. The task is to rotate the end effector from the initial frame into a goal frame without restrictions on the position. As before, the goal g is sampled uniformly in S^3 and then converted to other representations, and the action is limited to a relative rotation angle of at most 0.1π . During each episode, the agent can take 100 steps to reach the goal. We train the agent for 2,000,000 environment steps.

The results are shown on the bottom of Fig. 4. As before, we provide results for the average success rate, the final success rate, and the average reward per step of the resulting policy. For all results, we take the average of ten runs.

C. Pick-and-place task

Finally, we perform a pick-and-place task, where the robot arm needs to grasp and move a cube to a desired position and orientation. For this scenario, we use the same representation for state and action and showcase three orientation representations: Lie algebra m, rotation matrix SO(3), and positivereal-part quaternion S^{3+} . The robot arm is initialized in a single configuration. The cube's position is initialized in a plane on the ground, and its orientation is randomized and then projected to lie flat on the ground. The goal position is in the air, and the orientation is uniformly varied up to $\frac{\pi}{2}$ around a random axis from the initial orientation. The action is limited to a relative rotation angle of at most 0.2π . During each episode, the agent can take 100 steps to reach the goal. We train the agent for 10,000,000 environment steps. We use an expert policy to guide learning similar to the process described in [8].

The hardware setup is shown in Fig. 1 and the training results for three best representations, Lie algebra, rotation matrix, and positive-real-part quaternion, are shown in Fig. 6. For the results, we take the average of five runs.

We conduct two experiments on hardware. For the first experiment, we place the cube in the center of the table and use eight uniformly spaced starting orientations. The goals are sampled randomly, as in the simulation. We record four trials for each orientation. The second experiment has four different initial cube positions on a rectangle, and we use four uniformly spaced staring orientations. The goals are sampled



Fig. 6. Comparison of the training performance of selected orientation representations for the pick-and-place task with the same representation for state and action: Lie algebra m (blue), rotation matrices SO(3) (red), and positive-real-part quaternion S^+ (green).

randomly, as in the simulation. We record two trials for each orientation. The results for evaluation on hardware are shown in Tab. IV.

Experiment	Trials	Success
Center position	32	46.9%
Rectangle positions	32	50.0%

TABLE I	V
---------	---

SUCCESS RATES FOR PICK-AND-PLACE HARDWARE EXPERIMENTS

V. DISCUSSION

The results for direct and end effector orientation control, as well as picking-and-placing, show that, overall, the action representation has a larger effect on training performance, and the best training is achieved with our proposed Lie algebra actions, matching the theoretical motivation outlined in the paper. Rotation matrices $(SO(3) \text{ and } SO_{1:2}(3))$ and positive-real-part quaternions (S^{3+}) achieve good results, and the discontinuity in S^{3+} does not appear to deteriorate the performance. The results for the state representation are less conclusive. A potential reason could be that networks are able to extract relevant features from all representations, even without adhering to the theoretical manifold at the input. Nonetheless, Euler angles and quaternions perform consistently worse than other representations. A possible reason for this result could be the multi-cover of orientations in these representations. Using the Riemannian manifold actions proposed in [18] does not provide good performance. We refer to [19] for potential reasons.

The resulting policies with Lie algebra actions perform slightly worse than certain other representations despite their better training performance. We attribute this result to the fixed box-bound action scale, which results in a smaller maximum rotation in certain directions for Lie algebra actions compared to other representations. Accordingly, the final average reward is lower since it takes longer to reach the goal. Scaling the action dynamically to achieve the full range considerably deteriorates training performance, possibly because now the action scale also has to be learned.

Due to the computationally faster projection, the Lie algebra representation also has the lowest training and rollout

times. For the same reason, the Lie algebra representation is also faster than the Riemannian manifold action due to the simpler mapping from manifold to tangent space and back.

VI. CONCLUSIONS

We presented a novel state and action orientation representation for learning with neural networks in reinforcement learning based on the Lie algebra of orientations. Our results show that specifically using the Lie algebra actions results in better training performance and faster training and policy rollout, making them generally a suitable choice for learning with orientations in robotics. Our findings are consistent across tasks with increasing complexity, indicating that the results generally apply to different robotics settings.

ACKNOWLEDGMENT

The authors would like to thank Petar Bevanda for the technical discussions and his help in preparing the manuscript.

REFERENCES

- G. Brunner, B. Szebedy, S. Tanner, and R. Wattenhofer, "The urban last mile problem: Autonomous drone delivery to your balcony," in 2019 international conference on unmanned aircraft systems (icuas), pp. 1005–1012, IEEE, 2019.
- [2] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7512–7519, IEEE, 2021.
- [3] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, 2024.
- [4] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.
- [5] S. Yang, Z. Zhang, Z. Fu, and Z. Manchester, "Cerberus: Low-drift visual-inertial-leg odometry for agile locomotion," in 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 4193– 4199, IEEE, 2023.
- [6] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. Mc-Grew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal* of Robotics Research, vol. 39, no. 1, pp. 3–20, 2020.
- [7] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*, pp. 297–307, PMLR, 2022.
- [8] J. Brüdigam, A.-A. Abbas, M. Sorokin, K. Fang, B. Hung, M. Guru, S. Sosnowski, J. Wang, S. Hirche, and S. L. Cleac'h, "Jacta: A versatile planner for learning dexterous and whole-body manipulation," *arXiv* preprint arXiv:2408.01258, 2024.
- [9] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv* preprint arXiv:1711.02508, 2017.
- [10] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," arXiv preprint arXiv:1812.01537, 2018.
- [11] J. Ansel et al., "Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation," in Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, pp. 929–947, 2024.
- [12] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.
- [13] S. Kim and M. Kim, "Rotation representations and their conversions," *IEEE Access*, vol. 11, pp. 6682–6699, 2023.
- [14] 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," arXiv preprint arXiv:1509.02971, 2015.

- [15] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing* systems, vol. 30, 2017.
- [16] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5745–5753, 2019.
- [17] A. R. Geist, J. Frey, M. Zobro, A. Levina, and G. Martius, "Learning with 3d rotations, a hitchhiker's guide to so (3)," *arXiv preprint* arXiv:2404.11735, 2024.
- [18] N. Alhousani, M. Saveriano, I. Sevinc, T. Abdulkuddus, H. Kose, and F. J. Abu-Dakka, "Geometric reinforcement learning for robotic manipulation," *IEEE Access*, 2023.
- [19] N. Jaquier, L. Rozo, and T. Asfour, "Unraveling the single tangent space fallacy: An analysis and clarification for applying riemannian geometry in robot learning," in 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 242–249, IEEE, 2024.
- [20] N. Alhousani, H. Kose, and F. J. Abu-Dakka, "Reinforcement learning for orientation on the lie algebra," in 2023 31st Signal Processing and Communications Applications Conference (SIU), pp. 1–4, 2023.
- [21] M. Finzi, S. Stanton, P. Izmailov, and A. G. Wilson, "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data," in *International Conference on Machine Learning*, pp. 3165–3176, PMLR, 2020.
- [22] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 37–45, 2015.
- [23] R. Chakraborty, J. Bouza, J. H. Manton, and B. C. Vemuri, "Manifoldnet: A deep neural network for manifold-valued data with applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 799–810, 2020.
- [24] J. Li, Z. Yang, H. Liu, and D. Cai, "Deep rotation equivariant network," *Neurocomputing*, vol. 290, pp. 26–33, 2018.
- [25] D. García-Retuerta, R. Casado-Vara, A. Martin-del Rey, F. De la Prieta, J. Prieto, and J. M. Corchado, "Quaternion neural networks: Stateof-the-art and research challenges," in *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 456–467, Springer, 2020.
- [26] T. Parcollet, M. Morchid, and G. Linares, "A survey of quaternion neural networks," *Artificial Intelligence Review*, vol. 53, no. 4, pp. 2957– 2982, 2020.
- [27] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, 2009.
- [28] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in 2012 IEEE/RSJ international conference on intelligent robots and systems, pp. 5026–5033, IEEE, 2012.