2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

SE(3) Linear Parameter Varying Dynamical Systems for Globally Asymptotically Stable End-Effector Control

Sunan Sun* and Nadia Figueroa

***Abstract*— Linear Parameter Varying Dynamical Systems (LPV-DS) encode trajectories into an autonomous first-order DS that enables reactive responses to perturbations, while ensuring globally asymptotic stability at the target. However, the current LPV-DS framework is established on Euclidean data only and has not been applicable to broader robotic applications requiring pose control. In this paper we present an extension to the current LPV-DS framework, named Quaternion-DS, which efficiently learns a DS-based motion policy for orientation. Leveraging techniques from differential geometry and Riemannian statistics, our approach properly handles the non-Euclidean orientation data in quaternion space, enabling the integration with positional control, namely SE(3) LPV-DS, so that the synergistic behaviour within the full SE(3) pose is preserved. Through simulation and real robot experiments, we validate our method, demonstrating its ability to efficiently and accurately reproduce the original SE(3) trajectory while exhibiting strong robustness to perturbations in task space.**

I. INTRODUCTION

Adaptivity to new tasks and robustness to unforeseen perturbations and uncertainties are fundamental for safe integration of robots in human workspaces. On one hand, trajectory planning for position in $\mathbb{R}^d$ has significantly advanced and evolved into a multitude of approaches for different applications. These range from traditional path planning algorithms assuming a known environment and robot dynamics [1]–[3], to more adaptive and reactive alternatives using Dynamical System (DS) or Dynamic Movement Primitives (DMP) to encode complex trajectories [4]–[6]. On the other hand, trajectory planning for orientation in $\mathbb{S}^3$ requires more complex mathematical strategies due to the non-Euclidean nature of the orientation manifold and its multitude of representations such as Euler angles, rotation matrices and quaternions [7].

For many tasks it might be sufficient to control solely the position of the end-effector of a robot, however, a truly dexterous manipulator requires full end-effector pose control. While orientation controllers do exist, they often require hard-wiring and re-programming of every new task, not allowing the adaptability, flexibility and robustness to perturbations achieved by reactive DS position control strategies. Such discrepancy in development has hindered the capability of a fully interactive robot. In practice, the decoupling becomes common where the position and orientation are computed and executed via two independent controllers. The cost, however, is the loss of the dependency between position and orientation inherent to any task, let alone the inflexibility that comes with a non-adaptive orientation controller.

All authors are with the Department of Mechanical Engineering, University of Pennsylvania, Philadelphia PA 19104, USA
*Corresponding author. (e-mail: sunan@seas.upenn.edu)

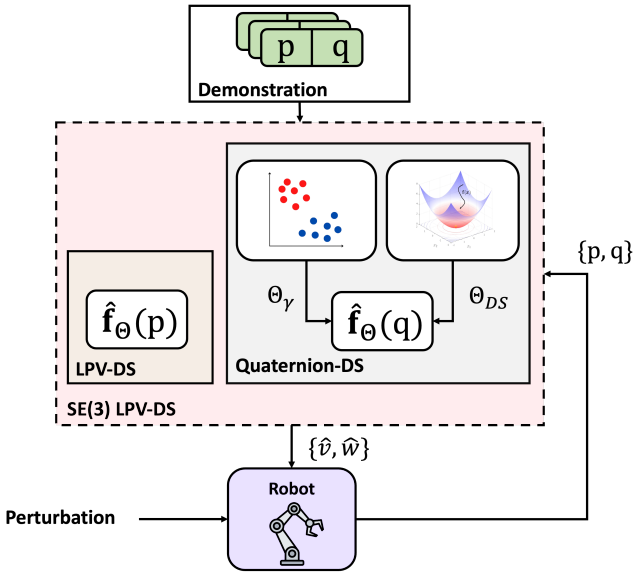

Fig. 1: The schematic of the **SE(3) LPV-DS** formulation which is composed of an ordinary LPV-DS [8] for position control and **Quaternion-DS** for orientation control; the architecture of Quaternion-DS consists of the clustering of the orientation trajectory and the optimization to minimize the prediction error; the resulting SE(3) LPV-DS takes the end-effector states: position $\mathbf{p}$ and orientation $\mathbf{q}$ as inputs, and generates the estimated desired linear velocity $\hat{v}$ and angular velocity $\hat{\omega}$, which are then passed down to command the robot via a low-level feedback controller.

Encoding trajectories as Dynamical System (DS) has been increasingly popular for enabling adaptive behaviour in robotic control [4]. DS-based motion policies leverage redundancy of solutions in dynamic environments and embed an infinite set of feasible solutions in a single control law to overcome external uncertainties and perturbations [9]. Among existing learning frameworks, recent neural network (NN) based formulations for stable DS motion policies show promising results in encoding highly non-linear trajectories; as adopting normalizing flows [10], euclideanizing flows [11] or via contrastive learning [12]; however, most works are mainly focused on Euclidean data and designing a NN on orientation manifold is not easily feasible due to its black box nature [13]. A recent work has proposed a diffeomorphism-based approach to learn stable motion policies on different Riemannian manifolds, including $\mathbb{R}^3 \times \mathbb{S}^3$ [14]. Yet, diffeomorphic and NN based approaches, while expressive, are computational and sample inefficient, as well as incapable of adapting to new task parameters or environments.

The Linear Parameter Varying Dynamical System (LPV-DS) formulation, on the other hand, is the seminal frame-

work in learning stable, time-independent DS-based motion policies from limited demonstrations [4], [8]. As opposed to NN-based learning that requires many trajectories and substantial computation time to reach stable solutions, LPV-DS is effective in learning trajectory behaviour with minimal data and higher computational efficiency for potential real-time incremental learning, as showcased in a recent extension capable of learning LPV-DS motion policies in seconds [15].

In addition, LPV-DS is comprised of a statistical model — a Gaussian Mixture Model (GMM) and a semi-definite optimization, making it an *explainable* pipeline. This enables greater flexibility in engineering specific responses [16], and allows for stability conditions constructed as constraints in the learning, offering a closed-form analytical solution to trajectory planning with theoretical guarantees such as stability and convergence. Furthermore, the LPV-DS framework grounded on trajectory data has shown to be able to generalize to new task instances in the Elastic-DS formulation [17].

In this paper we extend the current LPV-DS framework to encode SE(3) trajectories, referred to as **SE(3) LPV-DS**. In the light of the LPV-DS framework, we first introduce the **Quaternion-DS** for adaptive orientation control. With the proper handling and techniques from Riemannian statistics and differential geometry, the Quaternion-DS framework can generate a stable rotational motion policy given the orientation trajectories. We then formulate a comprehensive SE(3) LPV-DS framework by integrating the Quaternion-DS with the established LPV-DS method, enabling full SE(3) pose control using the translation and quaternion representation instead of the homogeneous matrix representation. We evaluate our approach through extensive empirical validation on real robot experiments. Our results demonstrate that the SE(3) LPV-DS maintains the intrinsic relationship between position and orientation, while preserving all the perks of LPV-DS including global asymptotic stability, strong robustness to external perturbations, and high computational efficiency.

Paper Organization: Section II introduces the mathematical preliminaries of the LPV-DS framework, Quaternion arithmetic and Riemannian statistics. The proposed methods are detailed in Section III and evaluated in Section V. Source code for the work is available at `https://github.com/SunannnSun/quaternion_ds`.

II. MATHEMATICAL PRELIMINARIES

A. LPV-DS formulation

We begin by introducing the existing LPV-DS framework for Euclidean data [8], which is typically used to encode position trajectories represented by the brown block in Figure 1. This will serve as the foundation for the formulation of the Quaternion-DS.

Let $\xi, \dot{\xi} \in \mathbb{R}^d$ represent the kinematic robot state and velocity vectors. In the DS-based motion policy literature [4], $\dot{\xi} = f(\xi)$ is a first-order DS that describes a motion policy in the robot's state space $\mathbb{R}^d$. The goal of DS-based learning from demonstration (LfD) is to infer $f(\xi) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ from data, such that any point ξ in the state space leads to a stable attractor $\xi^* \in \mathbb{R}^d$, with $f(\xi)$ described by a set of parameters Θ and attractor $\xi^* \in \mathbb{R}^d$; mathematically $\dot{\xi} = f(\xi; \Theta, \xi^*) \Rightarrow \lim_{t\to\infty} \|\xi - \xi^*\| = 0$, i.e., the DS is globally asymptotically stable (GAS) [18].

Learning $\dot{\xi} = f(\xi)$ can be framed as a regression problem, where the inputs are the state variables ξ and the outputs are the first-order time derivative $\dot{\xi}$. Such formulation gives rise to the utilization of statistical methods for estimating the parameters Θ. However, standard regression techniques cannot ensure globally asymptotic stability. To alleviate this, the LPV-DS approach was first introduced in the seminal work of [9] as a constrained Gaussian Mixture Regression (GMR) and then formalized as the untied GMM-based LPV-DS approach in [8], where a nonlinear DS is encoded as a mixture of continuous linear time-invariant (LTI) systems:

$$\begin{aligned} \dot{\xi} &= f(\xi; \Theta) = \sum_{k=1}^{K} \gamma_k(\xi) \left(\mathbf{A}_k \xi + b_k\right) \\ \text{s.t. } &\begin{cases} \left(\mathbf{A}_k\right)^T \mathbf{P} + \mathbf{P}\mathbf{A}_k = \mathbf{Q}_k, \mathbf{Q}_k = \left(\mathbf{Q}_k\right)^T \prec 0 \\ b_k = -\mathbf{A}_k \xi^* \end{cases} \end{aligned} \tag{1}$$

where $\gamma_k(\xi)$ is the state-dependent mixing function that quantifies the weight of each LTI system $(\mathbf{A}_k\xi + b_k)$ and $\Theta = \{\theta_\gamma\}_{\gamma=1}^K = \{\gamma_k, \mathbf{A}_k, b_k\}_{k=1}^K$ is the set of parameters to learn. The constraints of the Eq. 1 enforce GAS of the result DS derived from a parametrized Lyapunov function $V(\xi) = (\xi - \xi^*)^T \mathbf{P} (\xi - \xi^*)$ with $\mathbf{P} = \mathbf{P}^T \succ 0$ [4], [8].

To ensure GAS of Eq. 1, besides enforcing the Lyapunov stability constraints on the LTI parameters one must ensure that $0 < \gamma_k(\xi) < 1$ and $\sum_{k=1}^K \gamma_k(\xi) = 1 \ \forall \xi \in \mathbb{R}^d$. As noted in [8], this is achieved by formulating $\gamma_k(\xi) = \frac{\pi_k \mathcal{N}(\xi|\theta_k)}{\sum_{j=1} \pi_j \mathcal{N}(\xi|\theta_j)}$ as the *a posteriori probability* of the state ξ from a GMM used to partition the nonlinear DS into linear components. Here, K is the number of components corresponding to the number of LTIs, $\mathcal{N}(\xi|\theta_k)$ is the probability of observing ξ from the k-th Gaussian component parametrized by mean and covariance matrix $\theta_k = \{\mu_k, \mathbf{\Sigma}_k\}$, and π_k is the prior probability of an observation from this particular component satisfying $\sum_{k=1}^K \pi_k = 1$.

In [8] a two-step estimation framework was proposed to estimate the GMM parameters $\Theta_\gamma = \{\pi_k, \mu_k, \mathbf{\Sigma}_k\}_{k=1}^K$ and the DS parameters $\Theta_{DS} = \{\mathbf{A}_k, b_k\}_{k=1}^K$ forming $\Theta = \{\Theta_\gamma, \Theta_{DS}\}$. First, given the set of reference trajectories $\mathcal{D} := \{\xi_i^{\text{ref}} \ \dot{\xi}_i^{\text{ref}}\}_{i=1}^N$, where i is the sequence order of the sampled states, a GMM is fit to the position variables of the reference trajectory, $\{\xi_i^{ref} \ \}_{i=1}^N$, to obtain Θ_γ. The optimal number of Gaussians K and their placement can be estimated by model selection via Expectation-Maximization or via Bayesian non-parametric estimation. Then, Θ_{DS} are learned through a semi-definite program minimizing reproduction accuracy subject to stability constraints [4], [8].

B. Quaternion Arithmetic

The unit quaternion is a compact representation of orientation defined by $\mathbf{q} = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \in \mathbb{H} \subset \mathbb{R}^4$ with unit norm $\|\mathbf{q}\| = 1$. The inverse orientation is represented by its conjugate $\bar{\mathbf{q}} = w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}$. Both quaternions $\mathbf{q}$ and $-\mathbf{q} = -w - x\mathbf{i} - y\mathbf{j} - z\mathbf{k}$ represent the same orientation.

Given two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$, we can compute the displacement as $\Delta\mathbf{q} = \bar{\mathbf{q}}_1 \circ \mathbf{q}_2$ which is the quaternion product of $\bar{\mathbf{q}}_1$ and $\mathbf{q}_2$. The quaternion multiplication is not commutative; i.e. the order of multiplication matters and changing order will result in a different outcome. For example, the $\Delta\mathbf{q}$ computed above represents the amount of rotation required to rotate $\mathbf{q}_1$ onto $\mathbf{q}_2$ with respect to the body frame. If the time difference dt is provided, we can therefore compute the angular velocity by first converting the displacement in axis-angle representation as follows:

$$\omega = \frac{2}{dt} \arccos(w) \frac{(x, y, z)}{\|(x, y, z)\|} \tag{2}$$

Note again the computed ω is the angular velocity expressed in the body frame not the world frame due to the specific order of the quaternion multiplication.

C. Riemannian Manifold

Unit quaternions can also be conceptualized as residing on a three-dimensional hypersphere with a radius of 1, known as the 3-sphere or $\mathbb{S}^3$, which is embedded in four-dimensional Euclidean space $\mathbb{R}^4$. Recognizing the 3-sphere is a Riemannian manifold allows us to employ techniques from differential geometry and Riemannian statistics. First of all, a Riemannian manifold is a smooth manifold equipped with positive definite inner product defined in the tangent space at each point. This metric allows for the measurement of distances, angles, and other geometric properties on the manifold. Hereafter, we only consider the unit sphere as pertaining to the unit quaternions, and the provided formulas are mostly specific to the unit sphere. For clarity, we denote elements of the manifold in bold and elements in tangent space in fraktur typeface; i.e. $\mathbf{p} \in \mathcal{M}$ and $\mathfrak{q} \in T_{\mathbf{p}}\mathcal{M}$.

The notion of distance on the Riemannian manifold is a generalization of straight lines in Euclidean spaces. The minimum distance paths that lie on the curve, also called geodesics, are defined as $d(\mathbf{p}, \mathbf{q}) = \arccos(\mathbf{p}^T\mathbf{q})$ between two points on unit sphere, or $\mathbf{p}, \mathbf{q} \in \mathbb{S}^d$ [19], [20]. We can also compute the Riemannian equivalent of mean and covariance as follows,

$$\begin{aligned} \tilde{\mu} &= \underset{\mathbf{p} \in \mathbb{S}^d}{\operatorname{argmin}} \sum_{i=1}^{N} d(\mathbf{q}_i,\ \mathbf{p})^2 \\ \tilde{\mathbf{\Sigma}} &= \frac{1}{(N-1)} \sum_{i=1}^{N} \log_{\tilde{\mu}}(\mathbf{p}_i) \log_{\tilde{\mu}}(\mathbf{p}_i)^T. \end{aligned} \tag{3}$$

The average $\tilde{\mu}$, defined as the center of mass on unit sphere, employs the notion of the Fréchet mean [21], which extends the sample mean from $\mathbb{R}^d$ to Riemannian manifolds $\mathcal{M}$. In practice, $\tilde{\mu}$ can be efficiently computed in an iterative approach [22]. The empirical covariance $\tilde{\mathbf{\Sigma}}$ captures the dispersion of data in tangent space $T_{\mathbf{p}}\mathcal{M}$, where the logarithmic map $\log_{\mathbf{p}} : \mathcal{M} \to T_{\mathbf{p}}\mathcal{M}$ maps a point on the Riemannian manifold to the tangent space defined by the point of tangency $\mathbf{p}$ [22]–[24]:

$$\mathfrak{q} = \log_{\mathbf{p}}(\mathbf{q}) = d(\mathbf{p}, \mathbf{q}) \frac{\mathbf{q} - \mathbf{p}^T\mathbf{q}\mathbf{p}}{\|\mathbf{q} - \mathbf{p}^T\mathbf{q}\mathbf{p}\|}. \tag{4}$$

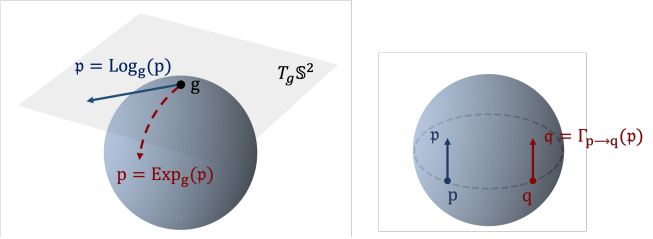

Fig. 2: Illustrative examples of the operations on Riemannian geometry: exponential/logarithmic mapping (left) and parallel transport (right), which are depicted on a $\mathbb{S}^2$ manifold embedded in $\mathbb{R}^3$

The inverse map is the exponential map $\exp_{\mathbf{p}} : T_{\mathbf{p}}\mathcal{M} \to \mathcal{M}$ which maps a point in tangent space of $\mathbf{p}$ to the manifold so that the mapped point lies in the direction of the geodesic starting at $\mathbf{p}$ [22]–[24]:

$$\mathbf{q} = \exp_{\mathbf{p}}(\mathfrak{p}) = \mathbf{p}\cos(\|\mathfrak{p}\|) + \frac{\mathfrak{p}}{\|\mathfrak{p}\|}\sin(\|\mathfrak{p}\|). \tag{5}$$

The parallel transport $\Gamma_{\mathbf{p}\to\mathbf{q}}(\mathfrak{p}) : T_{\mathbf{p}}\mathcal{M} \to T_{\mathbf{q}}\mathcal{M}$ transports a vector $\mathfrak{p}$ in the tangent space $T_{\mathbf{p}}\mathcal{M}$ at a point $\mathbf{p}$ on a manifold $\mathcal{M}$ to a vector in the tangent space $T_{\mathbf{q}}\mathcal{M}$ at point $\mathbf{q}$, along geodesic: [25]

$$\Gamma_{\mathbf{p}\to\mathbf{q}}(\mathfrak{p}) = \mathfrak{p} - \frac{\log_{\mathbf{p}}(\mathbf{q})^T\mathfrak{p}}{d(\mathbf{p}, \mathbf{q})^2}(\log_{\mathbf{p}}(\mathbf{q}) + \log_{\mathbf{q}}(\mathbf{p})). \tag{6}$$

The L2 norm $\|\log_{\mathbf{p}}(\mathbf{q})\|_2$ in the tangent space $T\mathbb{S}^d$ is equal to the geodesic distance between points $\mathbf{p}$ and $\mathbf{q}$ on the manifold: $d(\mathbf{p}, \mathbf{q})$. This provides an effective measure of deviations from a single point, allowing us to construct Gaussians on the manifold, as will be shown in Section III-C. However, this is true only when $\mathbf{p}$ is the point of tangency. In general, the inner product between two vectors in $T\mathbb{S}^d$ is not equal to the geodesic distance between their corresponding points in $\mathbb{S}^d$. [25]

III. QUATERNION DYNAMICAL SYSTEM

Section II-A introduces the general framework of LPV-DS that is only suitable to the Euclidean data such as position trajectories. Note that a learned DS via LPV-DS takes the form of $\dot{\xi} = f(\xi)$, and when the system is GAS, ξ will reach the target and $\dot{\xi}$ will diminish to zero over time. When it comes to orientation, we expect that the learned DS might take different forms with different inputs and outputs, but the objective should remain the same: i) the system can reach the target and ii) the system is stable at the target.

Among the various representations of orientations, quaternion stands out as compact and singularity-free. In accordance with the Euclidean LPV-DS, it is most intuitive to construct its orientation counterpart by taking quaternion as input and generating either angular velocity or the derivative of quaternion as output. However, neither angular velocity nor derivative of quaternion lives in quaternion space, meaning no direct affine mapping from quaternions to any of them.

Say we have a trajectory of unit quaternion, $\mathcal{Q} := \{\mathbf{q}^i\}_{i=1}^N$ and $\mathbf{q}_{att}$ is denoted as the attractor or target of this quaternion trajectory, we first project all the elements in the quaternion

trajectory to the tangent space defined by $\mathbf{q}_{att}$ via the logarithmic map in Eq. 4,

$$\mathfrak{q}_{att}^i = \log_{\mathbf{q}_{att}}(\mathbf{q}^i) \quad \forall\ i = 1,\ \ldots,\ N \tag{7}$$

where all the unit quaternions $\mathbf{q}^i \in \mathbb{S}^3$ have now been projected onto the tangent space defined by the attractor, $\mathfrak{q}_{att}^i \in T_{\mathbf{q}_{att}}\mathbb{S}^3$. Each projected vector represents the direction and distance from $\mathbf{q}_{att}$ to $\mathbf{q}^i$ along a geodesic path on the manifold while preserving the Riemannian metric of two vectors on the Riemannian manifold. We note the analogy of Eq. 7 to its Euclidean equivalent $(\xi - \xi^*)$ in Section II-A.

As we argue that neither angular velocity ω nor the derivative of quaternion $\dot{q}$ lives in quaternion space, we resolve to a discrete system that outputs the next desired orientation. Provided the angular velocity and time difference, we can integrate forward and compute:

$$(\mathbf{q}^i)^{des} = \mathbf{q}^i \circ (\omega^i \times dt), \tag{8}$$

where $(\omega^i \times dt)$ should be converted to the quaternion representation before being composed with the current quaternion. We then perform logarithmic map in Eq. 4, but this time w.r.t. the corresponding current state:

$$(\mathfrak{q}_{body}^i)^{des} = \log_{\mathbf{q}^i}(\mathbf{q}^i)^{des} \quad \forall\ i = 1,\ \ldots,\ N. \tag{9}$$

Via Eq. 9 obtain a new set of vectors representing the distance or displacement from every orientation to their desired state, and each vector resides in a unique tangent space defined by the current state $\mathbf{q}^i$. One can also draw the analogy between Eq. 9 and the linear velocity $\dot{\xi}$ in Section II-A. When the system is approaching the target, $\dot{\xi}$ should point towards the target and reach zero asymptotically. The discrete counterpart of velocity in quaternion space, or $\mathfrak{q}_{body}^i$ embodies the same idea as the displacement should lead to the target and gradually reduce to zero.

For mathematical completeness, we perform an additional step by parallel transporting each individual vector $\mathfrak{q}_{body}^i$ from their current state to the attractor as in Eq. 6:

$$(\mathfrak{q}_{att}^i)^{des} = \Gamma_{\mathbf{q}^i \to \mathbf{q}_{att}}(\mathfrak{q}_{body}^i)^{des} \tag{10}$$

so that both $\mathfrak{q}_{att}^i$, the vector from the attractor $\rightarrow$ current state, and $(\mathfrak{q}_{body}^i)^{des}$, the vector from the current state $\rightarrow$ desired state, are expressed in the same tangent space defined by the quaternion attractor $\mathbf{q}_{att}$.

Putting things together the quaternion-DS should bear the following form according to the LPV-DS framework in Eq. 1,

$$\begin{aligned} &(\hat{\mathfrak{q}}_{att}^i)^{des} = \sum_{k=1}^{K} \gamma_k(\mathbf{q}^i)\mathbf{A}_k \log_{\mathbf{q}_{att}} \mathbf{q}^i, \\ &\text{s.t. } \left\{ \ \sum_{j=1}^{K}\sum_{k=1}^{K} \gamma_j(\mathbf{q}^i)\gamma_k(\mathbf{q}^i)\mathbf{A}_j^T\mathbf{P}\mathbf{A}_k - \mathbf{P} \prec 0 \right. \end{aligned} \tag{11}$$

where $(\hat{\mathfrak{q}}_{att}^i)^{des}$ is the estimated desired orientation output by the DS, $\gamma_*(\mathbf{q}^i)$ is the state-dependent mixing function that quantifies the weight of each LTI system, K is the number of LTI systems partitioned by the statistical model, and $\mathbf{A} \in \mathbb{R}^{4\times4}$ is the linear parameters of each LTI system. The constraints enforce the globally asymptotic stability (GAS) at the target, which is derived from the Lyapunov stability conditions for a discrete system.

In the following sections, we will detail a) the stability analysis of the quaternion-DS, and how to b) convert the output of quaternion-DS to angular velocity for robot control, c) obtain the mixing functions $\gamma_*(\cdot)$ via statistical model, and d) learn the linear parameters $\mathbf{A}$ via optimization.

A. Stability analysis

To derive the Lyapunov constraints that enforce the GAS of our quaternion-DS as in Eq. 11, we choose the Lyapunov function to be quadratic in the tangent space defined by the attractor with $\mathbf{P} = \mathbf{P}^T \succ 0$ as,

$$V(\mathbf{q}^i, \mathbf{q}_{att}) = V(\mathfrak{q}_{q_{att}^i}^i) = \mathfrak{q}_{q_{att}^i}^{iT}\mathbf{P}\mathfrak{q}_{q_{att}^i}^i > 0, \tag{12}$$

Observe that for all $\mathbf{q}^i$ that is not $\mathbf{q}_{att}$, the Lyapunov function in Eq. 12 is always positive. According to the Lyapunov stability theory for discrete systems [26], the Lyapunov function must satisfy that its difference is always negative and only equal to zero at the target equilibrium:

$$\begin{aligned} \Delta V(\mathbf{q}^i) &= V(\mathbf{q}^{i+1}) - V(\mathbf{q}^i) \\ &= \mathfrak{q}_{q_{att}^{i+1}}^{iT}\mathbf{P}\mathfrak{q}_{q_{att}^{i+1}}^{i+1} - \mathfrak{q}_{q_{att}^i}^{iT}\mathbf{P}\mathfrak{q}_{q_{att}^i}^i \\ &= \mathfrak{q}_{q_{att}^i}^{iT}\left(\sum_{j=1}^{K}\sum_{k=1}^{K}\underbrace{\gamma_j(\mathbf{q}^i)\gamma_k(\mathbf{q}^i)}_{\succ 0}\mathbf{A}_j^T\mathbf{P}\mathbf{A}_k - \mathbf{P}\right)\mathfrak{q}_{q_{att}^i}^i \end{aligned} \tag{13}$$

Given that the mixing function $\gamma_*(\cdot)$ being positive and the matrix $\mathbf{P}$ being positive definite, we have to ensure that the summation of every terms within the parenthesis in Eq. 13 is negative definite. We note that when enforcing the linear parameter $\mathbf{A}_k$ of each LTI system to be negative definite: $\mathbf{A}_k \prec 0 \quad \forall k = 1,\ \ldots\ , K$, we are ensuring that

$$\begin{aligned} \mathbf{A}_k^T\mathbf{P}\mathbf{A}_k &\prec 0 \quad \forall k = j \\ \mathbf{A}_j^T\mathbf{P}\mathbf{A}_k &\preceq 0 \quad \forall k \neq j, \end{aligned} \tag{14}$$

and hence the difference in Eq. 13 is always negative except at the equilibrium.

When $\mathbf{q}^i$ reaches the target equilibrium $\mathbf{q}_{att}$, we have the Lyapunov function and its difference equal to zero,

$$\begin{aligned} V(\mathbf{q}_{att}, \mathbf{q}_{att}) &= \mathbf{0}^T\mathbf{P}\mathbf{0} = 0 \\ \Delta V(\mathbf{q}_{att}) &= \mathbf{0}^T\left(\sum_{j=1}^{K}\sum_{k=1}^{K}\gamma_j(\mathbf{q}^i)\gamma_k(\mathbf{q}^i)\mathbf{A}_j^T\mathbf{P}\mathbf{A}_k - \mathbf{P}\right)\mathbf{0} \\ &= 0. \end{aligned} \tag{15}$$

Hence, the Lyapunov function and its time difference satisfy all the necessary conditions to ensure the GAS of the quaternion-DS in the Lyapunov sense.

B. Conversion to Angular Velocity for Control

In a typical robotic application, the estimated desired state from the quaternion-DS formulation cannot be directly used by the low-level controller, and we need to recover the time derivative of the state space, that is the angular velocity in

this case. Provided that a time difference dt is known, we first parallel transport the estimated $(\hat{\mathbf{q}}^i_{att})^{des}$ from the attractor back to the current state $\mathbf{q}^i$,

$$(\hat{\mathbf{q}}^i_{body})^{des} = \Gamma_{\mathbf{q}_{att} \to \mathbf{q}^i}(\hat{\mathbf{q}}^i_{att})^{des}, \tag{16}$$

where the new vector is the estimated desired displacement expressed in the body frame. We then map this vector from the tangent space back to the quaternion space,

$$(\hat{\mathbf{q}}^i)^{des} = \exp_{\mathbf{q}^i}(\hat{\mathbf{q}}^i_{body})^{des}, \tag{17}$$

where the vector is the estimated desired orientation in quaternion space; and lastly compute the angular velocity given the current state and time difference,

$$\hat{\omega}^i = (\bar{\mathbf{q}}^i \circ (\hat{\mathbf{q}}^i)^{des})/dt. \tag{18}$$

The result of quaternion product needs to be converted into axis-angle representation before dividing it by dt to generate the proper angular velocity. Due to the specific order of the quaternion multiplication above, the computed ω is the estimated angular velocity w.r.t. the body frame, not the fixed frame. This ω can then be passed down to a low level controller such as a twist controller for a robot as in Fig. 1.

C. Quaternion Mixture Model

One advantage of LPV-DS is the use of statistical model to capture the intrinsic structure of trajectory data while accounting for the uncertainty within trajectory. This is manifested as the mixing function $\gamma(\cdot)$ in Eq. 11, which quantifies the weight of each LTI system and governs the transition between each LTI system at a given state.

Using GMM to cluster Euclidean data is straightforward as introduced in Section II-A where one can choose variants of GMM and inference methods to construct the mixing function. However, clustering quaternions using GMM is challenging because GMM assumes Euclidean metrics while quaternions are non-Euclidean. However, by recognizing that the unit quaternions reside on the Riemannian manifold or the 3-sphere, we can employ the techniques from differential geometry and apply the same treatment as in the previous Section III-A — projecting quaternion data onto the tangent space, locally approximating them as Euclidean vectors and fitting a GMM to the tangent vector. But caution is needed and we argue that the following quaternion mixture model is only useful and specific to our application and may not serve as a general framework for clustering sparse quaternions.

The orientation trajectory has a special trait that is missing from sparse quaternion data, that is the trajectory has an end point. Projecting unit quaternion data w.r.t. this end point bears physical meanings as we discussed in Section III; those tangent vectors represent the displacement from the target to the current state in terms of orientation. By clustering these tangent vectors in the tangent space, we are also implicitly segmenting the unit quaternion data in quaternion space. We argue that the tangent vectors that share similar displacement from the attractor should also be close to each other in the quaternion space, and this is only true when the tangent

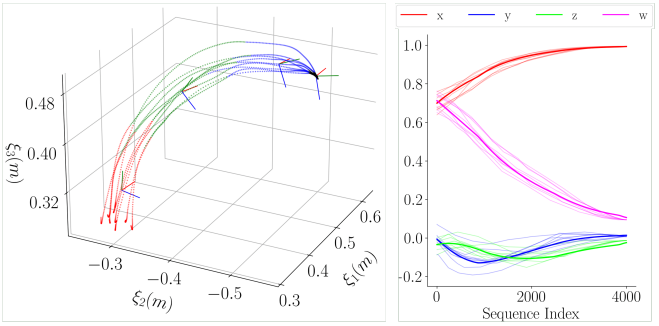

Fig. 3: The quaternion mixture model ($K = 4$) with the mean orientation of each Gaussian (left); note the color is only indicative of the clustering results on the quaternion trajectory, not position. The simulation result of quaternion-DS (right) shows the evolution of the quaternions in its 4 coordinates, where the thin lines are demonstration trajectory, and the thick lines are reproduction.

vector preserves the intrinsic geometry of the trajectory and Euclidean metrics is a fair approximation.

Another assumption underlying the LPV-DS framework is that trajectory should not *self-intersect*, given that an autonomous first-order DS system cannot handle higher-order derivatives nor sequential information. The constraint holds across various benchmark dataset for LPV-DS framework [8], [9], and is easy to enforce when data is collected in a continuous manner, as opposed to sparse points. Hence, this is a fair assumption for orientation data as well; i.e. no orientation trajectory should reach the same orientation more than once before stopping at the target. Inadvertently, this constraint also prevents the scenarios where a trajectory contains both $\mathbf{q}$ and $-\mathbf{q}$ which represent same orientation in task space while are drastically distinct vectors in quaternion space and its tangent space.

Say we have a trajectory of unit quaternion, $\mathcal{Q} = \{\mathbf{q}^i\}_{i=1}^N$, we project all its elements to the tangent space defined by $\mathbf{q}_{att}$ via logarithmic map in Eq. 4, and obtain the set of tangent vectors $\mathfrak{q}^i_{att}$ as in Eq. 7. When a GMM is fit to the tangent vectors, we can retrieve a label for each orientation data. Grouping the data points with the same assignment, we can then compute the mean $\tilde{\mu}$ and empirical covariance $\tilde{\mathbf{\Sigma}}$ as in Eq. 3. When computing the probability of observing a particular unit quaternion on a Gaussian:

$$\mathcal{N}(\log_{\tilde{\mu}}(\mathbf{q}^i)|\mu = \mathbf{0},\ \Sigma = \tilde{\mathbf{\Sigma}}), \tag{19}$$

note that the unit quaternion $\mathbf{q}$ is projected via logarithmic map w.r.t. the mean $\tilde{\mu}$, and the actual mean μ of the Gaussian is a zero vector as the logarithmic map of the mean w.r.t. itself is zero or the origin in tangent space.

We can then formulate the mixing function in Eq. 11 as the *a posteriori probability* of the quaternion $\mathbf{q}$ on GMM,

$$\gamma_k(\mathbf{q}) = \frac{\pi_k \mathcal{N}(\mathfrak{q}_{\tilde{\mu}_k}|\theta_k)}{\sum_{j=1} \pi_j \mathcal{N}(\mathfrak{q}_{\tilde{\mu}_j}|\theta_j)}, \tag{20}$$

where the quaternion is mapped w.r.t. the mean of each Gaussian component respectively, and the parameters of each Gaussian must contain a zero mean. Such formulation enforces that $0 < \gamma_k(\mathbf{q}) < 1$ and $\sum_{k=1}^K \gamma_k(\mathbf{q}) = 1\ \forall \mathbf{q} \in \mathbb{H}$, ensuring the GAS of Eq. 11 as discussed in Section III-A.

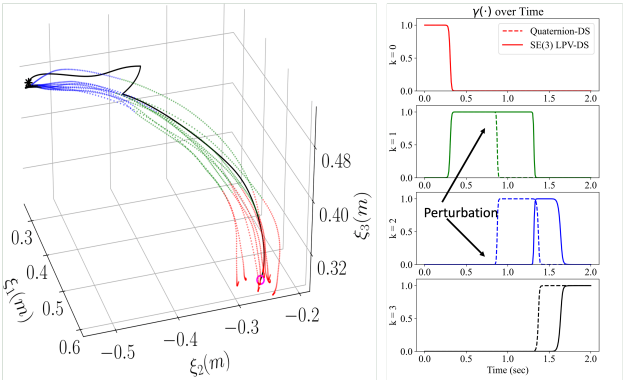

Fig. 4: The simulation results of SE(3) LPV-DS in the event of perturbation (left), where the demonstration data and clustering results are scatter points in color and the reproduction is the black curve; the value of mixing function $\gamma(\cdot)$ corresponding to each Gaussian ($K = 4$) during the simulation (right), where the solid lines are computed in SE(3) LPV-DS by Eq. 23, and the dashed lines are computed in Quaternion-DS by Eq. 20.

Note that the mixture model only partitions the original trajectory $\mathcal{Q} = \{\mathbf{q}^i\}_{i=1}^N$, leaving the negative trajectory $\{-\mathbf{q}^i\}_{i=1}^N$ in the other half of the quaternion space unmodeled. We can mirror each Gaussian components by inverting the means $\tilde{\mu}_k$, hence doubling the Gaussian components and covering the entire quaternion space.

D. Optimization

For mathematical completeness, we present the semi-definite optimization formulation that learns the linear parameters $\mathbf{A}_k$ of each LTI system. Following the same principle in the previous sections, the optimization is performed in the tangent space by minimizing the L2 norm of the error between the reference desired orientation and the prediction,

$$\begin{aligned} \underset{\Theta}{\text{minimize}} \quad & \sum_{i=1}^{N} \left\| (\hat{\mathbf{q}}_{att}^i)^{des} - (\mathbf{q}_{att}^i)^{des} \right\|^2 \\ \text{subject to} \quad & \mathbf{A}_k \prec 0, \quad \forall k = 1, \dots, K, \end{aligned} \tag{21}$$

where the reference desired orientation is computed by Eq. 7, the prediction is computed by Eq. 11 and Eq. 20, and the constraint comes from the stability analysis in Section III-A. In Fig. 3, we illustrate a learned quaternion-DS.

IV. SE(3) LPV-DS for Pose Control

While Quaternion-DS offers a comprehensive framework for encoding orientation trajectory and generating rotational motion policy in quaternion space, running Quaternion-DS with any Euclidean LPV-DS concurrently does not solve the missing synergy within the pose as discussed in Section I.

SE(3) LPV-DS is an extension of the Quaternion-DS by introducing an implicit dependency between position and orientation. Previously, either in Euclidean LPV-DS or Quaternion-DS, the GMMs partition the trajectory in either $\mathbb{R}^3$ or $\mathbb{S}^3$, and the formulated mixing function $\gamma(\cdot)$ only accounts for the LTI systems in their respective vector space as discussed in Section II-A and Section III-C. On the other hand, SE(3) LPV-DS concatenates the position and the projected orientation as an augmented input, $[\mathbf{p},\ \mathbf{q}_{att}]^T \in \mathbb{R}^3 \times T\mathbb{S}^3$, and fits a GMM to the augmented input in the combined vector space. This is possible due to the construction of Quaternion-DS where the tangent vectors provide an effective measure in deviation w.r.t. the attractor by preserving the geodesic between points on the manifold when the point of tangency is the attractor. Similar to the Section III-C, we can recover the label of each input data from the clustering results and construct the corresponding Gaussians with the probability as follows,

$$\begin{aligned} \mathcal{N}\left(\begin{bmatrix} \mathbf{p} \\ \log_{\tilde{\mu}}(\mathbf{q}) \end{bmatrix} \middle| \ \mu = \begin{bmatrix} \mu_{\mathbf{p}} \\ \mathbf{0} \end{bmatrix}, \right. \\ \left. \Sigma = \frac{1}{N-1} \sum_{i=1}^{N} \begin{bmatrix} \mathbf{p} - \mu_{\mathbf{p}} \\ \log_{\tilde{\mu}}(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \mathbf{p} - \mu_{\mathbf{p}} \\ \log_{\tilde{\mu}}(\mathbf{q}) \end{bmatrix}^T \right), \end{aligned} \tag{22}$$

where the mean of the Gaussian is the positional mean concatenated with a zero vector, $\mathbf{0} \in \mathbb{R}^4$, and the deviation from mean is formulated by combining the distance in Euclidean space $\mathbf{p} - \mu_{\mathbf{p}}$ and the geodesic on the manifold $\log_{\tilde{\mu}}(\mathbf{q}_i)$. Note when computing the probability of a given pose, the quaternion component of the input needs to be transformed via logarithmic map w.r.t. the mean $\tilde{\mu}$ while the position part of the state remains unchanged. This gives rise to the mixing function $\gamma(\cdot)$ with the following form,

$$\gamma_k \left(\begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \right) = \frac{\pi_k \mathcal{N}\left(\begin{bmatrix} \mathbf{p} \\ \log_{\tilde{\mu}_k}(\mathbf{q}) \end{bmatrix} \middle| \ \theta_k \right)}{\sum_{j=1} \pi_j \mathcal{N}\left(\begin{bmatrix} \mathbf{p} \\ \log_{\tilde{\mu}_j}(\mathbf{q}) \end{bmatrix} \middle| \ \theta_j \right)}, \tag{23}$$

where the parameters of each Gaussian θ should take the form of Eq. 22. This formulation of $\gamma(\cdot)$ allows the SE(3) LPV-DS to account for both position and orientation while partitioning the trajectory into Gaussian components and learning the linear parameters $\mathbf{A}$ during optimization. When reproducing the trajectory at any give state, the SE(3) LPV-DS assigns value to each $\gamma(\cdot)$ according to the pose rather than position or orientation only.

Fig. 4 presents a comparison result between the SE(3) LPV-DS and Quaternion-DS when responding to perturbation. The perturbation only occurs in position, and is manually added when a robotic system (e.g. end effector of a robot arm) is transitioning from the second LTI system (green) to the third LTI system (blue). We illustrate the response by plotting the value of mixing function $\gamma(\cdot)$ over time, which represents the weight each LTI system receives at a given state; e.g. a value of $\gamma_k = 1$ means that the robotic system is fully governed by the k-th LTI system. We observe that when the perturbation occurs at $t = 0.8s$, the Quaternion-DS alone is completely unaware and proceeds to transition to the next LTI (blue). On the other hand, the SE(3) LPV-DS delays the transition and remains in its current LTI (green) until the position is rectified.

By fitting the mixture model to the pose trajectory and learning a single DS in the combined vector space, our approach coordinates position and orientation together in a

coupled manner. By contrast, previous works, e.g. *Neural Ordinary Differential Equation solvers* (NODEs) [27] and *contracting dynamical system primitives* (CDSP) [28], learn separate autonomous nonlinear DS for position and orientation. To the best of our knowledge, SE(3) LPV-DS is the first DS-based motion policy that preserves the synergy inherent to any task while guaranteeing robustness to perturbations.

V. EXPERIMENT

A. Benchmark Comparison

Baseline: We compared our approach against the baseline method — *Neural Ordinary Differential Equation solvers (NODEs) which has been recently exploited to learn vector fields in trajectory-based learning [27]*. With an equivalent parameter size, NODEs converges significantly faster than its neural-based predecessors, e.g. the *Imitation Flow* (iFlow) [29], while still being able to accurately encode trajectory including both position and orientation.

Dataset: We evaluated our approach on the *RoboTasks9* dataset from [27]. The dataset of 9 real-world tasks includes box opening, plate stacking and etc., in which both the position and orientation of the robot's end-effector vary over time. Each task, provided kinsethetically by humans, contains 9 trajectories with a size of 1000 observations each including both position $\mathbf{p} \in \mathbb{R}^3$ and orientation $\mathbf{q} \in \mathbb{H}$.

Metrics: To assess the learning performance, we report the widely used metrics: *Dynamic Time Warping error* (DTW) for end-effector position and *Quaternion error* [30] for orientation. In addition, we measure the model complexity and computation time; both are crucial for continual and incremental learning in real-time scenario.

Results: The model complexity in terms of parameter size for NODEs is pre-defined by the layers of neural networks and remains fixed regardless of trajectory size. By contrast, our approach requires more LTI systems with more parameters to represent an increasing trajectory. *Computation time* for NODEs is largely determined by the iterations as opposed to our approach growing linearly with data size. Nevertheless, our approach is formulated as a semi-definite optimization that can be solved in near real-time, requiring significantly fewer parameters and less time for convergence than NODEs by an order of magnitude as shown in Fig. 5.

When assessing the *reproduction accuracy*, we report three different scenarios: starting from initial points, starting from unmodeled regions, and reacting to perturbations. We notice that when starting from the provided initial points, both approaches reproduced the trajectory with low mean error and small variance. However, when starting from *outside* the provided initial points, our approach still managed to maintain a comparable accuracy (low error in both position and quaternion) while the baseline experienced a few trials of divergence and instability resulting in larger variance and upper extremes. Similar results occur to the added perturbation: our model, though suffers a slight increase in error, ensures a low variance and prevents any upper outlier. On the other hand, the baseline failed to generalize a stable vector field outside the unmodeled region, leading to worse performance in both position and orientation.

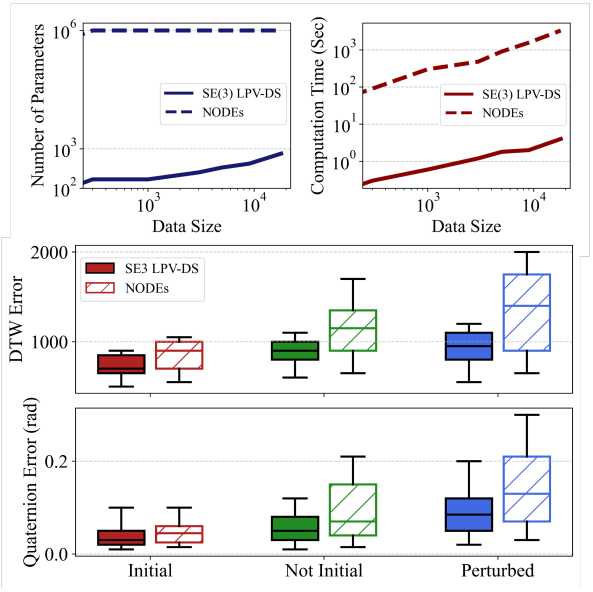

Fig. 5: Comparison of parameter size and computation time with data size (top), and DTW error and quaternion error (lower is better) over 9 tasks in three scenarios. Solid bars represent our approach, and hatched bars depict the baseline. Bars range from lower quartile to upper quartile, with whiskers indicating extreme values.

B. Real Robot Experiments

We validate our approach on four exemplary real-world tasks including *box opening*, *book shelving*, *water pouring* and *plate moving*. Each task contains 3 demonstrations including position and orientation with a total size of approximately 2000 observations. In Figure. 6, we show the sequence of snapshots of a robot successfully performing the four tasks, as well as the evolution of position and quaternion over time. We further illustrate its robustness to perturbations and ability to generalize to the unmodeled region in the supplementary video.

VI. CONCLUSIONS

In this paper, we present an extension to the current LPV-DS framework, enabling the learning of trajectory planning/control on orientation and full pose trajectory. Comparing against the baseline methods on real-world tasks, we verified that our approach achieves comparable results on reproduction accuracy and generalization ability while maintaining an efficient model complexity and computation efficiency. However, we note that the underlying assumption of LPV-DS requires no *self-intersecting* trajectory, excluding the broader range of tasks in real life. This leads to the future work where the learned Dynamical System should incorporate higher-order derivative with sequential information.

REFERENCES

[1] A. Ude, "Trajectory generation from noisy positions of object features for teaching robot paths," *Robotics and Autonomous Systems*, vol. 11, no. 2, pp. 113–127, 1993.

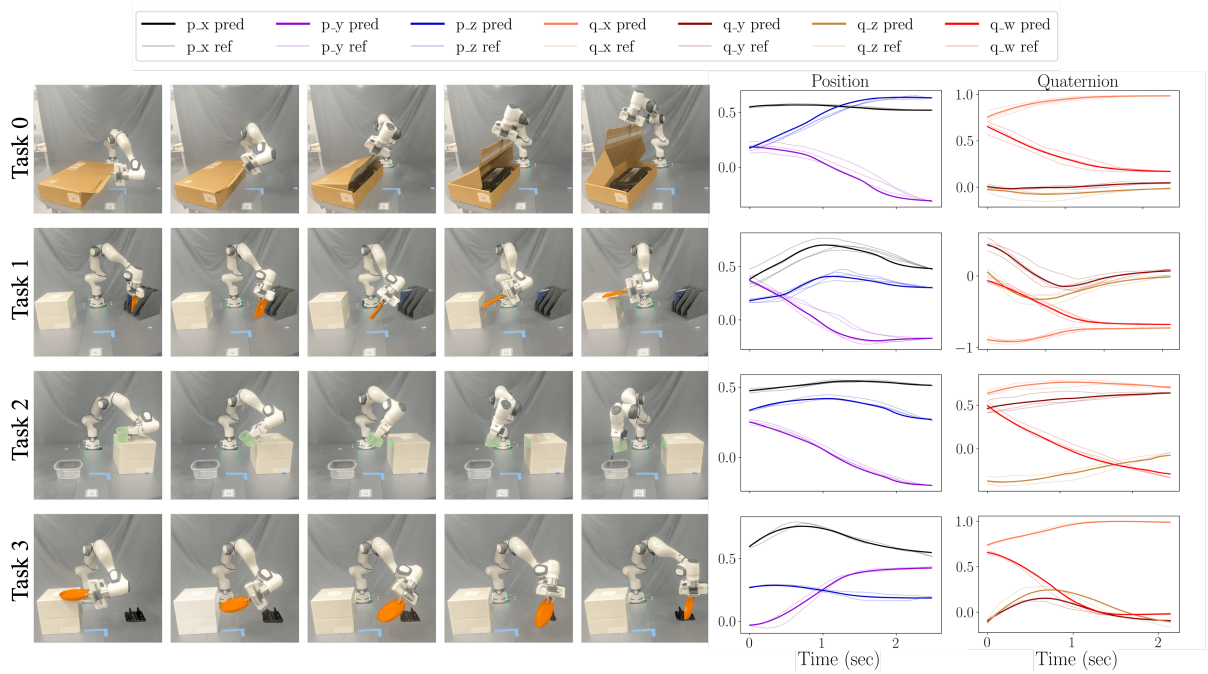

Fig. 6: A robot trained via SE(3) LPV-DS to perform four different real-world tasks: 0) *box opening*, 1) *book shelving*, 2) *water pouring*, 3) *plate moving*. The training trajectories in which both position and orientation vary over time are provided by kinesthetic demonstrations. For each task, we illustrate the time evolution of the reproduced trajectory — position and quaternion in their respective coordinates.

[2] J.-H. Hwang, R. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," vol. 2, 11 2003, pp. 1444 – 1449 vol.2.

[3] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 409–413, 2006.

[4] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. The MIT Press, 2022.

[5] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[6] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research. The Eleventh International Symposium: With 303 Figures*. Springer, 2005, pp. 561–572.

[7] R. Campa and H. de la Torre, "Pose control of robot manipulators using different orientation representations: A comparative review," in *2009 American Control Conference*, 2009, pp. 2855–2860.

[8] N. Figueroa and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," in *Proc. of The 2nd Conference on Robot Learning*, vol. 87. PMLR, 2018, pp. 927–948.

[9] S. M. Khansari-Zadeh and A. Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[10] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *2020 IEEE/RSJ IROS*, 2020, pp. 5231–5237.

[11] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff, "Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems," in *Proc. of the 2nd Conference on Learning for Dynamics and Control*, vol. 120. PMLR, Jun 2020, pp. 630–639.

[12] R. Pérez-Dattari and J. Kober, "Stable motion primitives via imitation and contrastive learning," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3909–3928, 2023.

[13] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[14] J. Zhang, H. B. Mohammadi, and L. Rozo, "Learning riemannian stable dynamical systems via diffeomorphisms," in *6th Annual Conference on Robot Learning*, 2022.

[15] S. Sun, H. Gao, T. Li, and N. Figueroa, "Directionality-aware mixture model parallel sampling for efficient linear parameter varying dynamical system learning," 2023.

[16] N. Figueroa and A. Billard, "Locally active globally stable dynamical systems: Theory, learning, and experiments," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 312–347, 2022.

[17] T. Li and N. Figueroa, "Task generalization with stability guarantees via elastic dynamical system motion policies," in *7th Annual Conference on Robot Learning*, 2023.

[18] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, the book can be consulted by contacting: PH-AID: Wallet, Lionel.

[19] M. do Carmo, *Riemannian Geometry*, ser. Mathematics (Boston, Mass.). Birkhäuser, 1992.

[20] J. Lee, *Introduction to Riemannian Manifolds*, ser. Graduate Texts in Mathematics. Springer International Publishing, 2019.

[21] M. Arnaudon, F. Barbaresco, and L. Yang, "Medians and means in riemannian geometry: Existence, uniqueness and computation," 11 2011.

[22] X. Pennec, "Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements," *Journal of Mathematical Imaging and Vision*, vol. 25, pp. 127–154, 07 2006.

[23] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on riemannian manifolds," *IEEE RA-L*, vol. 2, no. 3, pp. 1240–1247, 2017.

[24] S. Calinon, "Gaussians on riemannian manifolds: Applications for robot learning and adaptive control," *IEEE Robotics & Automation Magazine*, vol. 27, no. 2, pp. 33–45, 2020.

[25] M. P. Do Carmo and J. Flaherty Francis, *Riemannian geometry*. Springer, 1992, vol. 2.

[26] N. Bof, R. Carli, and L. Schenato, "Lyapunov theory for discrete time systems," 2018.

[27] S. Auddy, J. Hollenstein, M. Saveriano, A. Rodríguez-Sánchez, and J. Piater, "Scalable and efficient continual learning from demonstration via a hypernetwork-generated stable dynamics model," 2024.

[28] H. C. Ravichandar and A. Dani, "Learning position and orientation dynamics from demonstrations via contraction analysis," *Autonomous Robots*, vol. 43, no. 4, pp. 897–912, 2019.

[29] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," 2020.

[30] M. Saveriano, F. Franzel, and D. Lee, "Merging position and orientation motion primitives," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019.