IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Behavior-Controllable Stable Dynamics Models on Riemannian Configuration Manifolds

Byeongho Lee*[1], Yonghyeon Lee*[2], Junsu Ha*[3], and Frank C. Park[3]

***Abstract*—Due to their stability and robustness properties, stable dynamical systems (SDSs) have received considerable attention as a means of representing motions in learning from demonstration tasks. Designing vector fields that fit complex trajectories while ensuring stability still remains a key challenge; although recent deep-learning-based methods have shown substantial progress in this direction, their tendency to overfit to demonstration trajectories often leads to undesirable behaviors, particularly as tasks deviate from demonstrations. At a fundamental level, the only reliable way to address this lack of generalization is to provide supervision in out-of-demonstration regions. Focusing on two types of general behaviors, mimicking and contracting, we propose a behavior-controllable stable dynamics model (BCSDM), a one-parameter family of SDS that allows users to adjust the system's overall behavior depending on user intent. We show how to extend the BCSDM to accommodate demonstrations of multiple tasks, and also propose a deep operator vector field for memory-efficient encoding of multiple dynamical systems. Extensive experiments on tasks that involve mimicking or contracting behaviors demonstrate the advantages of BCSDMs over existing state-of-the-art SDS learning methods.**

I. INTRODUCTION

In the dynamical systems approach to learning from demonstration (LfD), typically one is given a set of demonstrations in the form of reference trajectories, and the goal is to design a differential equation on a robot's configuration space that drives the system toward these reference trajectories [1]. As a concrete example, given a desired reference trajectory $x_d(\tau) \in \mathbb{R}^n$, where $\tau \in [0, T]$, one seeks to construct a mapping $f : \mathbb{R}^n \to \mathbb{R}^n$ (f is called a *vector field*) such that any solution trajectory $x(t)$ to $\dot{x} = f(x)$ approaches $x_d(\tau)$ in the limit as t goes to infinity. An example of a stable dynamical system is shown in Fig. 1(a); the arrows represent velocity vectors at various points in the configuration space.

An ideal dynamical systems-based LfD method should be able to produce a vector field that fits complex reference trajectories, and converges stably and in a well-behaved manner to the reference trajectory, all while being robust to external perturbations. Because the space of feasible vector fields is infinite-dimensional, existing dynamical systems-based methods for LfD have focused on developing more computationally tractable finite-dimensional methods. These methods aim to learn a parametric vector field that converges to the reference trajectory, with the final point of the trajectory acting as a globally asymptotically stable equilibrium.

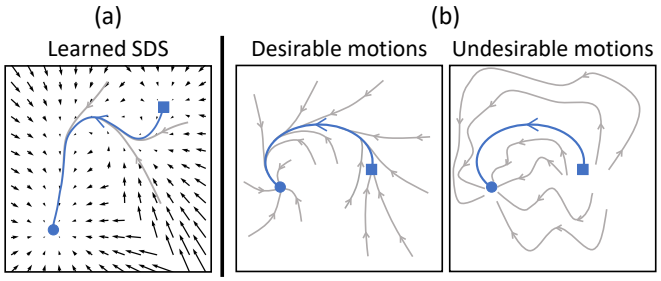

Fig. 1. A vector field and phase portraits of learned stable dynamical systems (SDS). Blue circular points represent stable equilibrium points, while black arrows represent output vector fields. Reference trajectories are shown in blue; solution trajectories to the vector fields are shown in gray. (a) A learned vector field that contracts toward the reference trajectory. (b) Phase portraits of two different dynamical systems that converge toward the same reference trajectory. The system on the left exhibits desirable behavior in out-of-demonstration regions, whereas the system on the right generates undesirable jerky motions in these regions.

Earlier works in this direction typically utilize simple parametric models, e.g., Gaussian mixture models, to learn a vector field that fits a given reference trajectory [2-7]. These methods ensure that the generated vector fields are stable, either by mapping a known stable vector field to a target configuration space, or by constructing a vector field that conforms to a pre-specified Lyapunov function. Although stability can be guaranteed in both cases, the expressivity of the vector field can become highly restricted, resulting in vector fields that are unable to fit complex reference trajectories, or are poorly behaved away from the reference trajectory.

More recent methods have tried to exploit deep learning models that can generate more expressive vector fields. By exploiting Lyapunov functions based on input convex neural networks [8], or diffeomorphic mappings based on normalizing flows [9], methods such as [10-15] can generate vector fields that better fit complex reference trajectories while still guaranteeing stability.

The main drawback with these more recent deep learning-based methods is a general lack of robustness: in regions that are distant from the demonstration data (or *out-of-demonstration regions*) the generated vector fields can produce jerky and erratic motions; see Fig. 1(b) for an illustration of desirable and undesirable motions generated by a vector field. Although deep learning models are highly expressive, because

This work was supported in part by IITP-MSIT under Grant RS-2021-II212068 (SNU AI Innovation Hub), Grant 2022-220480, and Grant RS-2022-II220480 (Training and Inference Methods for Goal Oriented AI Agents); in part by the MSIT Global Research Support Program in the Digital Field under Grant RS-2024-00436680 supervised by IITP; in part by KIAT under Grant P0020536 (HRD Program for Industrial Innovation); in part by MOTIR under Grant RS-2025-25460896 and Grant RS-2025-25462891; in part by SNU-AIIS, SNU-IPAI, SNU Institute for Engineering Research, and Microsoft Research Asia under Grant; in part by SRRC NRF under Grant RS-2023-00208052; and in part by AP092701. Yonghyeon Lee was the beneficiary of an individual grant from CAINS supported by a KIAS Individual Grant (AP092701) via the Center for AI and Natural Sciences at Korea Institute for Advanced Study. [1]Future Robotics AI Group, Samsung Electronics, Seoul 06765, South Korea. [2]Department of Mechanical Engineering, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA (e-mail: yhl@mit.edu). [3]Department of Mechanical Engineering, Seoul National University, Seoul 08826, South Korea (e-mail: fcp@snu.ac.kr). *Equal contribution. Corresponding author: F. C. Park (e-mail: fcp@snu.ac.kr).

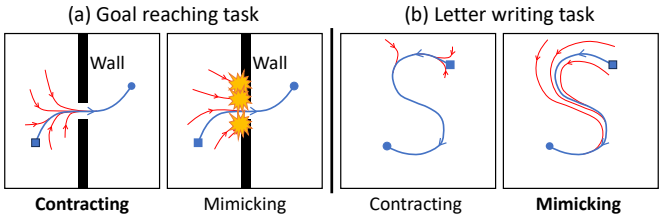

Fig. 2. Two tasks that require different behaviors. The blue curves represent demonstration trajectories, while the red curves depict the solution trajectories of the stable dynamical systems. In (a), contracting behavior is desired to navigate through the narrow passage, whereas mimicking behavior results in a collision with the wall. In (b), mimicking is preferred to preserve the shapes of the letters, while contracting behavior alters the shape of the written letter.

the reference trajectory covers only a small portion of the configuration space, deep learning-based models can generate erratic vector fields in unsupervised regions.

On the other hand, there is only so much that can be inferred or designed from a limited number of demonstrations. It would be unfair to attribute the erratic motions generated for out-of-demonstration regions as fundamental flaws of existing algorithms. We believe the only plausible remedy is to provide more supervision; the key question is how to do so in an effective and intuitive way, with minimal intervention from the user.

The key idea proposed in this paper is a means of providing supervision in out-of-demonstration regions, based on the idea that tasks can be broadly divided into those for which rapid convergence to the reference trajectory is more important, and those for which mimicking the overall shape of the reference trajectory is more important. Fig. 2 illustrates these two tasks. In Fig. 2(a), the robot must rapidly contract toward the demonstration trajectory in order to pass through the narrow passage and successfully complete the task. In contrast, for the letter-writing task of Fig. 2(b), it is more important for the robot to preserve the shape of the letter 'S' rather than to contract as quickly as possible to the demonstration trajectory, especially if the robot is initially in a distant out-of-demonstration region.

The task of Fig. 2(a) is an example of what we call **contracting** behavior, while the task illustrated in Fig. 2(b) is an example of **mimicking** behavior. The qualitative importance of many tasks can in practice be described as lying between the two extremes of contracting and mimicking, or in some cases even smoothly varying between these two behaviors throughout various stages of the motion. The rate of convergence to the demonstration trajectory is what distinguishes between mimicking and contracting behaviors, or any behaviors that lie in-between.

The main contribution of this paper is a one-parameter family of stable dynamical systems, called *Behavior-Controllable Stable Dynamics Models (BCSDMs)*, that allow the user to provide supervision in out-of-demonstration regions based on this qualitative distinction between contracting and mimicking behaviors. The user can, by appropriately adjusting the convergence rate, generate vector fields that can balance between contracting and mimicking (see Fig. 3). One of the key technical ideas behind our realization of BCSDMs is to define velocity labels in out-of-demonstration regions for two orthogonal directions, the *contracting direction* and the *parallel direction*. We show that any linear combination of velocities along these orthogonal directions results in a globally stable dynamical system. The resulting dynamical systems are also able to fit complex trajectories while behaving predictably and as desired in out-of-demonstration regions.

A second key feature of our BCSDM framework is that it admits curved configuration spaces. In practice, many robots have curved configuration spaces, e.g., robots with kinematic constraints (for example, two robots grasping and manipulating a common object, or a robot tracing a curve on a curved surface like a sphere), or robots whose task spaces involve the Special Euclidean group SE(3) or any of its subgroups. Whereas most existing dynamical systems-based LfD frameworks are realized explicitly in Euclidean space, in this paper we adopt the Riemannian framework, and consider configuration spaces that can be modeled as a Riemannian manifold. With the Riemannian framework, the orthogonal velocity directions that represent mimicking and contracting have natural counterparts that can be defined using geodesics and parallel transport. In this paper we provide an explicit algorithmic realization of these geometric concepts.

We also introduce a memory-efficient model capable of encoding multiple vector fields simultaneously; this is done by extending BCSDMs to accommodate multiple demonstration trajectories, each corresponding to a different task. This involves, among other things, learning a mapping that associates a trajectory with a vector field, i.e., an operator that maps one function to another. Taking advantage of the recent deep operator learning method [16] for identifying differential equations, we propose a corresponding Deep Operator Vector Field (DeepOVec) framework, in which a single DeepOVec model can generate appropriate vector fields for various target demonstration trajectories.

To validate our approach, we conduct two experiments that involve learning stable dynamical systems on curved configuration spaces: (i) a letter-writing task on the two-sphere S^2, and (ii) an umbrella-inserting task in the space of rigid body motions SE(3). The letter-writing task places an emphasis on mimicking behavior, while the umbrella-inserting task emphasizes contracting behavior. We show that our BCSDM and DeepOVec frameworks can be successfully employed for both tasks, leading to superior performance in terms of fitting accuracy, mimicking and contracting behavior, and overall task success rate compared to existing state-of-the-art methods.

II. RELATED WORK

In this section, we first review the existing literature on various motion representations in learning from demonstration, highlighting the advantages of autonomous first-order dynamical systems (i.e., time-invariant velocity fields), which we aim to construct in this paper. We then discuss recent advances in learning-based approaches for constructing stable and robust dynamical systems. Next, given our focus on velocity fields on Riemannian manifolds, we introduce existing methods for designing neural network-based velocity fields in

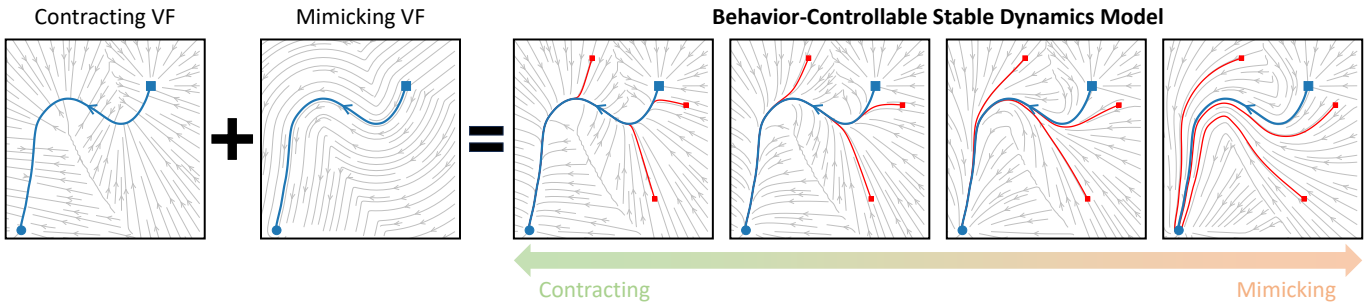

Fig. 3. Streamline plots of various BCSDMs. BCSDM combines contracting and mimicking velocities, of which the behavior is smoothly controllable from contracting to mimicking.

this setting. Finally, we survey prior works that attempt to combine contraction and mimicking behavior – similar to our approach – and highlight the key technical differences and relative applicability of these methods compared to ours.

A. Motion Representations in Learning from Demonstration

The literature on Learning from Demonstration (LfD) has developed various mathematical models to represent demonstration trajectories, broadly categorized into two types: (i) probabilistic distributions of trajectories, using techniques such as Gaussian mixtures, Gaussian processes, and autoencoders [17-29], and (ii) dynamical systems whose solutions fit the demonstration trajectories [2-5,10,30-47]. Dynamical systems-based methods can be further subdivided into Dynamic Movement Primitives (DMP) [35] and Stable Dynamical Systems (SDS) [2]. Both systems ensure stability, but differ in their approaches: DMPs use time-dependent acceleration fields with learnable force terms, while SDS utilize time-independent velocity fields.

In practical applications, trajectory distribution-based approaches typically involve sampling a trajectory followed by low-level tracking control, such as PD control. However, this setup lacks explicit mechanisms for handling deviations caused by external forces or perturbations, which can lead to undesirable or even extreme outcomes if the perturbations are significant. DMP variants, while effective in many scenarios, often struggle with synchronization issues when subjected to perturbations. Due to their time-dependent nature, they may fail to maintain proper alignment between the task's progress and the world clock, frequently resulting in unintended trajectories.

In contrast, SDS approaches exhibit notable robustness to both spatial and temporal perturbations. By assigning desired velocities in a globally time-invariant manner across the entire configuration space, they can effectively adapt to disturbances. In this paper, we build upon this research direction to develop a novel dynamical system that allows users to flexibly generate desired behaviors according to task requirements, controlling the balance between contracting and mimicking behaviors.

B. Methods for Stability Guarantees in Stable Dynamical Systems

The primary issue in the SDS learning literature has been the development of velocity field models that can fit complex trajectories while maintaining stability. The strategies to guarantee the stability of these systems can be roughly divided into three categories: (i) Lyapunov function-based [2,7,10,30,41,43,47,48], (ii) contraction theory-based [5,42,44-46,49-54], and (iii) diffeomorphism-based [11,13-15,55-57]. Lyapunov function-based approaches project the output of the velocity field such that a specific Lyapunov function – either pre-specified or learned simultaneously – always decreases, ensuring asymptotic convergence to its minimum point. Contraction theory-based approaches attempt to make the system contractive [51] – either by using additional regularization terms or by employing specialized models – so that the distance between neighboring solution trajectories decreases exponentially over time. Diffeomorphism-based approaches involve learning a diffeomorphism that maps a simple, stable reference velocity field to one that accurately fits a given demonstration trajectory.

These methods generally involve trade-offs between model expressivity – how complex a trajectory they can fit – and computational complexity. Adding additional structures, such as the convexity requirement for the Lyapunov function, the contraction property of a dynamical system, or the invertibility of the diffeomorphism, does not come for free; it requires sophisticated parametric models or neural networks. While some implementations use relatively cheap computational methods, they often struggle to fit complex trajectories. On the other hand, using deep neural networks with special architectures to capture complex trajectories significantly increases computational cost, often leading to instability in training and prolonged learning times.

In this paper, we take an orthogonal approach. Rather than using neural networks with specialized structures, we propose a non-parametric method that guarantees the almost-global stability property. We do use neural networks to fit the non-parametric field to address resolution issues arising from sparse demonstration data points. However, we rely only on simple fully connected neural networks with basic fitting losses. Although stability is not theoretically guaranteed, we present empirical evidence suggesting that our system is sufficiently stable, highlighting its practical usability. Further details can be found in Section IV.

C. Stable Dynamical Systems on Riemannian Manifolds

Constructing Stable Dynamical Systems (SDS) on Riemannian manifolds presents complex challenges. Yet, several frameworks have been proposed on Riemannian manifolds.

Among Lyapunov function-based approaches, Sun et al. [58] extend LPV-DS [7] to SE(3) by geometrically defining the Lyapunov function in the tangent space of SO(3). Some diffeomorphism-based methods geometrically construct a neural network-parameterized diffeomorphism and express it either in a global coordinate chart [14] or in dynamic local charts [15]. Mohammadi et al. [46] propose a contraction theory-based method on Riemannian manifolds, formulated in a global coordinate chart. Distinct from methods with formal stability guarantees, Perez et al. [57] encourage stability by adding a geodesic distance-based loss term on the manifold.

Our framework differs from these works in several respects. First and foremost, prior methods emphasize stability guarantees while paying limited attention to behavior in out-of-demonstration regions; in contrast, our framework explicitly prescribes behavior in such regions, yielding intuitive, well-behaved motions. Second, several works handle geometric challenges on curved manifolds using local, global, or dynamic charts [14,15,46]; these chart parameterizations are susceptible to distortion. Our formulation is coordinate-free, and the network operates directly in the ambient space, reducing chart-induced distortion. Lastly, our geometric construction is distinctive: we define a mimicking velocity via parallel transport. In our system, this yields local distance preservation to the reference trajectory – a property we prove – and it is central to our theorem; to our knowledge, this use has not appeared in similar contexts.

D. Vector Fields that Combine Contracting and Mimicking Velocities

The idea of combining two different vector fields – one contracting toward the reference trajectory and the other directing orthogonally, in some sense parallel to the trajectory velocity direction – with varying weights has been explored in a few existing works. This concept has been referred to as *contour control* and *guiding vector fields*. In this section, we review these approaches, address their limitations, and explain the significance of our method in this context. Additionally, we review an SDS framework – LAGS-DS – that is conceptually related yet distinct from this idea [47].

Contour control [1,59] algorithms define time-dependent vector fields by adding a corrective velocity component toward the reference trajectory – i.e., reducing contour error – to a time-varying trajectory tracking velocity. Each term can, in some sense, be considered as contracting and mimicking components, analogous to our approach. We note that the time-dependent nature of these vector fields can make them sensitive to spatial and temporal disturbances, as discussed in Section II-A. A more notable limitation is that their implementations assume Euclidean spaces. For general curved spaces, e.g., a Riemannian configuration manifold, one may use local coordinates to apply these methods. However, due to geometric distortions in the coordinate space, the contracting and mimicking effects no longer function meaningfully.

Recent literature includes guiding vector field methods [60-62] that can be applied to Riemannian manifolds and are often used for path following for UAVs. The key idea is to represent the reference trajectory in an m-dimensional manifold as the intersection of zero-level sets of $m-1$ potential functions. The contracting velocity is then defined in the direction that decreases all potential functions, while the mimicking velocity is defined in the direction that preserves them. However, there are two major challenges. First, designing appropriate potential functions for a complex reference trajectory is difficult and may even be infeasible. Second, the geometric meaning of the two directions is ambiguous. These directions are a result of potential function design rather than being derived from first principles. This contrasts with the ideal case considered in this paper, where the contracting direction should decrease the distance to the trajectory in the steepest manner, and the mimicking direction should preserve the distance to the trajectory.

While not a fusion of mimicking and contracting, it is worth noting that, among stable dynamical systems design frameworks, LAGS-DS combines two different dynamics [47] – one for direct convergence to the equilibrium and the other for contraction toward the reference trajectory. However, LAGS-DS differs in that it does not enforce mimicking behavior in out-of-demonstration regions. Additionally, it shares one of the aforementioned limitations – namely, the assumption of Euclidean space.

In contrast to these works, we design the contracting and mimicking velocity directions based on geometric principles, leveraging operations in Riemannian geometry such as geodesics, parallel transport, and related concepts – naturally defined on Riemannian manifolds in a coordinate-invariant way. This approach allows us to determine these directions easily, regardless of the complexity of the reference trajectory. As a result, the contracting velocity in our framework directly reduces the geodesic distance to the reference trajectory, while the mimicking velocity preserves the distance to the reference trajectory.

III. GEOMETRIC PRELIMINARIES

The Behavior-Controllable Stable Dynamics Models (BCSDM) framework in Section IV involves the implementation of the contracting direction and mimicking direction on Riemannian configuration manifolds. Given an input and the reference trajectory, we first find the point on the trajectory closest to the input. The contracting direction is then defined to be approaching the closest point taking the shortest path, and the parallel direction is defined by moving the velocity of the trajectory at the closest point parallel to itself.

In Euclidean space, these definitions are straightforward. Since the shortest path is a straight line, the contracting direction becomes $x^* - x$. Also, the parallel direction is simply defined as $\dot{x}^*$. However, on Riemannian configuration manifolds, such definitions are invalid, where vectors defined in the same way do not lie on the tangent space of x. To properly define these velocities to be confined in the tangent space, the concepts of shortest path on manifolds (geodesic) and of transporting vectors between tangent spaces (parallel transport) are exploited [63]. In this section, we focus on manifolds in

Euclidean space, introducing the concepts of geodesics and parallel transport. We provide formulas for the two-sphere and manifold of 3×3 rotation matrices, foundational for our later experiments.

A. Manifolds in Euclidean space

In this paper, we consider a differentiable manifold $\mathcal{M}$ of dimension m embedded in $\mathbb{R}^n$ that can be globally parametrized in *implicit* form as follows:

$$\mathcal{M} = \{x \in \mathbb{R}^n \,|\, g(x) = 0\}, \tag{1}$$

where $g : \mathbb{R}^n \to \mathbb{R}^{n-m}$ and $\frac{\partial g}{\partial x}(x) \in \mathbb{R}^{(n-m)\times n}$ is of maximal rank $n-m$ at all $x \in \mathcal{M}$. Then the tangent space attached at $x \in \mathcal{M}$, denoted by $T_x\mathcal{M}$, can be defined as the m-dimensional linear subspace in $\mathbb{R}^n$ as follows:

$$T_x\mathcal{M} := \{v \in \mathbb{R}^n \,|\, \frac{\partial g}{\partial x}(x)v = 0\}, \tag{2}$$

which is the null space of the $(n-m)\times n$ matrix $\frac{\partial g}{\partial x}(x)$.

To lay the groundwork for introducing parallel transport later, we introduce a linear orthogonal projection map $\pi : \mathbb{R}^n \to T_x\mathcal{M}$. Let v be an arbitrary vector in $\mathbb{R}^n$. Then the projection map π maps v as follows:

$$\pi(v) = \Big(I - \frac{\partial g}{\partial x}^T \Big(\frac{\partial g}{\partial x}\frac{\partial g}{\partial x}^T\Big)^{-1}\frac{\partial g}{\partial x}\Big)v, \tag{3}$$

where I is the $n\times n$ identity matrix. We note that $\pi^2(v) = \pi(v)$ and $\pi(v) \in T_x\mathcal{M}$.

Example Consider the set of three-dimensional unit vectors

$$S^2 := \{u \in \mathbb{R}^3 \,|\, u^T u = 1\}, \tag{4}$$

which forms a two-dimensional manifold. This manifold is globally implicitly parametrized by $g(u) = u^T u - 1$. The tangent space is given by $T_u S^2 = \{v \in \mathbb{R}^3 \,|\, u^T v = 0\}$ and the projection map is $\pi(v) = (I - uu^T)v$ for $v \in \mathbb{R}^3$.

Example Consider the set of 3×3 rotation matrices

$$\mathrm{SO}(3) := \{R \in \mathbb{R}^{3\times 3} \,|\, R^T R = I, \det R = 1\}, \tag{5}$$

which forms a three-dimensional manifold. This manifold is globally implicitly parametrized by $g(R) = R^T R - I$, where the determinant of R is positive. The tangent space

$$T_R\mathrm{SO}(3) = \{V \in \mathbb{R}^{3\times 3} \,|\, V = R\Omega\}, \tag{6}$$

where Ω is any 3×3 skew symmetric matrix, and the projection map is as follows [64]:

$$\pi(V) = \frac{V - RV^T R}{2} \text{ for } V \in \mathbb{R}^{3\times 3}. \tag{7}$$

B. Riemannian Metrics and Geodesics

Manifolds in Euclidean space can be naturally equipped with Riemannian metrics by leveraging the Euclidean geometry of $\mathbb{R}^n$. Specifically, let $v_1, v_2 \in T_x\mathcal{M}$. From an extrinsic view, note that these can be regarded as elements in $\mathbb{R}^n$. Then the inner product in $T_x\mathcal{M}$, denoted by $\langle\cdot,\cdot\rangle_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$, can be defined as the Euclidean inner product $\langle v_1, v_2\rangle = v_1^T v_2$. The family of inner products $\langle\cdot,\cdot\rangle_x$ for $x \in \mathcal{M}$ is called a *Riemannian metric*. We denote the *Riemannian norm* of $v \in T_x\mathcal{M}$ by $\|v\|_x = \sqrt{\langle v, v\rangle_x}$. Throughout, we ignore the subscript x since we use a metric that is Euclidean in the ambient space.

The Riemannian metric gives rise to the notion of geodesics, the shortest-path curves in $\mathcal{M}$. Let $x_1, x_2 \in \mathcal{M}$ and consider a curve $\gamma : [0,1] \to \mathcal{M}$ such that $\gamma(0) = x_1$ and $\gamma(1) = x_2$. A *geodesic curve* between x_1 and x_2 is a uniform-speed curve that minimizes its length:

$$\mathrm{Len}(\gamma) = \int_0^1 \sqrt{\langle\gamma'(s), \gamma'(s)\rangle}ds,$$

where $\gamma'(s) := \frac{d\gamma}{ds}(s)$. Since the geodesic γ is a uniform speed-curve and the time interval is 1, $\mathrm{Len}(\gamma) = \|\gamma'(s)\| \times 1$ for any $s \in [0,1]$. In addition, we note that its energy functional $E(\gamma) := \int_0^1 \langle\gamma'(s), \gamma'(s)\rangle\, ds = (\mathrm{Len}(\gamma))^2$.

Example The Riemannian metric for S^2 in Euclidean space $\mathbb{R}^3$, i.e., $\langle v_1, v_2\rangle_u = v_1^T v_2$ for $u \in S^2$, leads to the following geodesic curve between $u_1, u_2 \in S^2$:

$$\gamma(s) = \cos(\theta s)u_1 + \sin(\theta s)\frac{u_2 - (u_1^T u_2)u_1}{\|u_2 - (u_1^T u_2)u_1\|}, \tag{8}$$

where $\theta = \arccos(u_1^T u_2)$ is the geodesic distance between u_1 and u_2.

Example The Riemannian metric for SO(3) in Euclidean space $\mathbb{R}^{3\times 3}$, i.e., $\langle V_1, V_2\rangle_R = \frac{1}{2}\mathrm{Tr}(V_1^T V_2)$ for $R \in \mathrm{SO}(3)$, leads to the following geodesic curve between $R_1, R_2 \in \mathrm{SO}(3)$:

$$\gamma(s) = R_1 \exp(s[w]), \tag{9}$$

where $\exp(\cdot)$ and $\log(\cdot)$ are matrix exponential and logarithm [65] and $[w] = \log(R_1^T R_2)$.

C. Exponential Map

In Euclidean space, a particle $x \in \mathbb{R}^m$ with velocity $v \in \mathbb{R}^m$ moves to $x+v$ after a unit of time. However, on a Riemannian manifold $\mathcal{M}$, given a point $x \in \mathcal{M}$ and a velocity vector $v \in T_x\mathcal{M}$, the operation $x+v$ typically results in a point that does not lie on the manifold. The *(Riemannian) exponential map* $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ is introduced, providing a way to map a tangent vector v to a corresponding point on the manifold. Specifically, $\exp_x(v)$ is the endpoint of a geodesic starting at x with an initial velocity v, where the geodesic's length is equal to the norm $\|v\|$, i.e., $\exp_x(v) := \gamma(1)$ where γ is a geodesic curve such that $\gamma(0) = x$ and $\dot\gamma(0) = v$. Throughout, the Riemannian exponential map will be consistently denoted with the subscript $\exp_x(\cdot)$, distinguishing it from the conventional exponential function $\exp(\cdot)$.

The exponential map is crucial for discrete integration of differential equations on Riemannian manifolds[1]. For a vector

[1]More generally, any first-order retraction map – a smooth mapping $R_x : T_x\mathcal{M} \to \mathcal{M}$ that satisfies $R_x(0) = x$ and approximates the exponential map to first order near the origin of $T_x\mathcal{M}$ – can be used to project v onto the manifold.

field $f : \mathcal{M} \to T\mathcal{M}$, a Euclidean-like discrete-time system, $x_{t+1} = x_t + f(x_t)\Delta t$, where Δt is the time step, may escape the manifold. To ensure x_{t+1} lies on $\mathcal{M}$, the system can be defined using the exponential map:

$$x_{t+1} = \exp_{x_t}(f(x_t)\Delta t),$$

guaranteeing x_{t+1} remains on $\mathcal{M}$.

Example Given $u \in S^2$ and $v \in T_u S^2$, the exponential map is as follows:

$$\exp_u(v) = \exp([u \times v])u, \tag{10}$$

where $[v]$ is the skew-symmetric matrix of a vector $v \in \mathbb{R}^3$ and $\exp(\cdot)$ is matrix exponential.

Example Given $R \in \mathrm{SO}(3)$ and $V \in T_R\mathrm{SO}(3)$, the exponential map is as follows:

$$\exp_R V = R\ \exp(\Omega), \tag{11}$$

where $\exp(\cdot)$ is matrix exponential and $\Omega = R^T V$.

D. Parallel Transport

Parallel transport refers to the method of moving tangent vectors from one point on a manifold to another along a curve. In flat manifolds, these vectors remain unchanged during parallel transport. However, in manifolds with greater *intrinsic* curvature, the changes to the vectors become more pronounced.

We first introduce the *covariant derivative*, a formal way to measure how a vector changes as you move along a path on a curved manifold, accounting for the manifold's *intrinsic* curvature. Imagine walking along a surface like a sphere while carrying an arrow that represents a vector. As you move, you want to understand how this arrow changes direction relative to the surface itself – not just in the ambient Euclidean space. The covariant derivative adjusts the standard notion of a derivative by subtracting out the effects of the manifold's curvature, allowing you to focus on the intrinsic change of the vector within the manifold.

Specifically, consider a curve $x(s) \in \mathcal{M}$ and a velocity vector assigned to each point of the curve, denoted by $v(s) \in T_{x(s)}\mathcal{M}$. The covariant derivative of $v(s)$ along the tangent vector direction of the curve $x(s)$ is

$$\frac{D}{ds}v(s) := \pi(\frac{d}{ds}v(s)), \tag{12}$$

where $\pi : \mathbb{R}^n \to T_{x_t}\mathcal{M}$ is a linear orthogonal projection map. $\frac{D}{ds}v(s)$ keeps the derivative $\frac{d}{ds}v(s)$ within the tangent space, that measures the rate at which $v(s)$ changes *in the manifold*.

The parallel transport of a tangent vector $v \in T_{x_0}\mathcal{M}$ to another location $x_s \in \mathcal{M}$ along a curve $\gamma(s)$ such that $\gamma(0) = x_0$ and $\gamma(s) = x_s$ is a tangent vector in $T_{x_s}\mathcal{M}$. Denoting this transported vector by $v(s) \in T_{x_s}\mathcal{M}$, $v(s)$ is a solution of the following ordinary differential equation:

$$\frac{D}{ds}v(s) = 0. \tag{13}$$

An important insight here is that in curved manifolds, the result of parallel transporting a vector depends on the specific path taken between two points.

Throughout, we consider parallel transports along geodesic curves and denote the parallel transport of a tangent vector $v \in T_{x_1}\mathcal{M}$ to $x_2 \in \mathcal{M}$ with a mapping

$$\Gamma_{x_1}^{x_2} : T_{x_1}\mathcal{M} \to T_{x_2}\mathcal{M} \quad \text{s.t.} \quad v \mapsto \Gamma_{x_1}^{x_2}(v). \tag{14}$$

We note that, in general, parallel transport from x_1 to x_2 and then from x_2 to x_3 differs from parallel transport directly from x_1 to x_3, i.e., $\Gamma_{x_2}^{x_3}(\Gamma_{x_1}^{x_2}v) \neq \Gamma_{x_1}^{x_3}(v)$.

Example The parallel transport of $v \in T_{u_1}S^2$ to $u_2 \in S^2$ is

$$\Gamma_{u_1}^{u_2}(v) = \mathrm{Rot}(u_1 \times u_2, \arccos(u_1^T u_2))v, \tag{15}$$

where $\times$ is the cross product and $\mathrm{Rot}(w, \theta)$ is the 3×3 rotation matrix around w-axis for θ.

Example The parallel transport of $V \in T_{R_1}\mathrm{SO}(3)$ to $R_2 \in \mathrm{SO}(3)$ is as follows [64]:

$$\Gamma_{R_1}^{R_2}(V) = R_1 \exp(\frac{1}{2}[w])R_1^T V \exp(\frac{1}{2}[w]), \tag{16}$$

where $[w] = \log(R_1^T R_2)$.

IV. BEHAVIOR-CONTROLLABLE STABLE DYNAMICS MODELS

We begin this section with the notation and assumptions used throughout. We consider an m-dimensional Riemannian configuration manifold, $\mathcal{M}$, embedded in Euclidean space $\mathbb{R}^n$. We use an implicit parametrization of $\mathcal{M}$, denote a configuration by $x \in \mathcal{M} \subset \mathbb{R}^n$, and denote the Riemannian metric of $\mathcal{M}$ by $\langle \cdot, \cdot \rangle$. We denote a demonstration trajectory by $x_d(\tau) \in \mathcal{M}$ for $\tau \in [0, T]$. All demonstration trajectories are assumed to be not self-intersecting and to have zero terminal velocities.

In Sections IV-A, IV-B, and IV-C, we assume a single demonstration trajectory, $x_d(\tau)$, and propose a one-parameter family of stable dynamical systems. Each system is designed to fit $x_d(\tau)$, and its behavior can be adjusted by users through changes to the parameter. We then extend this concept to scenarios involving multiple demonstration trajectories with different terminal configurations in Section IV-D.

Specifically, in Section IV-A, we design two non-parametric vector fields: the *locally* contracting and *locally* mimicking vector fields. These fields either contract toward or locally mimic $x_d(\tau)$. Additionally, we introduce a *local* guidance vector field, defined as a weighted sum of the two. These fields are defined locally because, as discussed later, there exist regions where the vector changes discontinuously, making a global definition challenging. In Section IV-B, we leverage the local guidance vector field to define a discrete-time dynamical system. Under certain mild assumptions, we prove that this system is globally asymptotically stable at a goal configuration. In Section IV-C, to address computational challenges in non-parametric vector fields, we propose a method for fitting parametric models, which we refer to as Behavior-Controllable Stable Dynamics Models (BCSDMs). Lastly, in Section IV-D,

inspired by the recent deep operator learning literature [16], we propose a Deep Operator Vector Field (DeepOVec) that encodes multiple BCSDMs in a memory-efficient way.

A. Locally Contracting, Locally Mimicking, and Local Guidance Vector Fields

Let x be a point in $\mathcal{M}$, and let x^* be the nearest configuration to x in $\{x_d(\tau) \mid \tau \in [0,T]\}$, with the assumption that there exists a unique minimal geodesic between x and x^*. We denote this geodesic curve by $\gamma_x(s)$, noting that $\gamma_x(0) = x$ and $\gamma_x(1) = x^*$. Let $\mathcal{U} \subset \mathcal{M}$ be an open subset containing x such that, for all $x' \in \mathcal{U}$, there exists a unique nearest configuration to x' in $\{x_d(\tau) \mid \tau \in [0,T]\}$ and a unique geodesic connecting them. We denote $\dot{x}^* = \frac{dx_d}{d\tau}(\tau^*)$, where $x^* = x_d(\tau^*)$.

We first provide the definition of a **Locally Contracting Vector Field (CVF)**:

Definition 1: Let $\gamma_x(s)$ be a geodesic curve connecting x and the nearest configuration $x^* \in \{x_d(\tau) \mid \tau \in [0,T]\}$, a vector field in $\mathcal{U}$ defined as

$$\mathrm{CVF}(x; x_d) := \frac{d\gamma_x}{ds}(0) \in T_x\mathcal{U} \tag{17}$$

is said to be a locally contracting vector field to $x_d(\tau)$.

First, we can observe that a particle following $\dot{x} = \mathrm{CVF}(x; x_d)$ consistently moves towards the nearest configuration, x^*, on the demonstration trajectory $x_d(\tau)$. As a result, the distance to the trajectory, denoted by $D(x)$, continuously decreases:

Proposition 1: Let $x(t)$ be a local solution of $\dot{x} = \mathrm{CVF}(x; x_d)$ for $t \in (-\varepsilon, \varepsilon)$ such that $x(0) = x$. Then $\dot{D}(x) = -D(x) \leq 0$, where the equality holds when x is on $\{x_d(\tau) \mid \tau \in [0,T]\}$.

Proof: See Appendix A.

Second, the norm of the velocity equals the geodesic distance, i.e., $\|\mathrm{CVF}(x; x_d)\| = \mathrm{Len}(\gamma_x)$ (see the preliminary Section III-B). This means that the farther a particle is from the trajectory, the faster it approaches the trajectory. Once x reaches $x_d(\tau)$, $\mathrm{CVF}(x; x_d)$ becomes zero.

We now provide the definition of a **Locally Mimicking Vector Field (MVF)**:

Definition 2: Let $\gamma_x(s)$ be a geodesic curve connecting x and the nearest configuration $x^* \in \{x_d(\tau) \mid \tau \in [0,T]\}$, $\dot{x}^* = \frac{dx_d}{dt}(\tau^*)$ such that $x^* = x_d(\tau^*)$, and $\Gamma^x_{x^*} : T_{x^*}\mathcal{M} \to T_x\mathcal{M}$ be a parallel transport map (see Section III-D). A vector field in $\mathcal{M}$ defined as

$$\mathrm{MVF}(x; x_d) := \Gamma^x_{x^*}(\dot{x}^*) \in T_x\mathcal{U} \tag{18}$$

is said to be a locally mimicking vector field to $x_d(\tau)$.

First, we describe how $\mathrm{MVF}(x; x_d)$ functions as a mimicking field. The term 'mimicking' may not be as intuitive as 'contracting'. In this paper, we consider a trajectory to be mimicking $x_d(\tau)$ if, at any point x on the trajectory, the velocity vector $\dot{x} \in T_x\mathcal{M}$ is *similar* to $\dot{x}^* \in T_{x^*}\mathcal{M}$, where x^* is the nearest configuration on $x_d(\tau)$ to x. In the Euclidean configuration space, $\mathrm{MVF}(x; x_d) = \dot{x}^*$, thus the solution trajectory of $\dot{x} = \mathrm{MVF}(x; x_d)$ perfectly mimics $x_d(\tau)$ according to our definition of a mimicking trajectory.

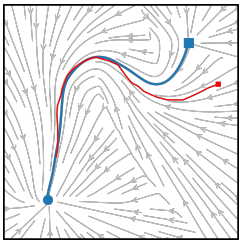

Fig. 4. Example of η tuning with an additional demonstration. The blue trajectory is the original demonstration, and the red trajectory is newly provided by a human expert for tuning. η is optimized to best fit the new demonstration, with streamlines showing the underlying vector field.

On Riemannian configuration manifolds, however, the varying tangent spaces along the manifold mean that $\dot{x}^* \in T_{x^*}\mathcal{M}$ may not live in another tangent space $T_x\mathcal{M}$ and thus $\|\dot{x}^* - \dot{x}\|$ for $\dot{x} \in T_x\mathcal{M}$ can no longer be considered as the similarity metric between $\dot{x}^*$ and $\dot{x}$. Instead, we consider them *similar* if one can be parallel transported to the other along the geodesic curve, since by definition this results in minimal changes in the vectors (refer to Section III-D). Consequently, $\mathrm{MVF}(x; x_d) \in T_x\mathcal{M}$ is the velocity vector that most closely matches $\dot{x}^* \in T_{x^*}\mathcal{M}$, thereby generating solution trajectories that effectively mimic $x_d(\tau)$.

Second, we observe that the distance from a particle following $\dot{x} = \mathrm{MVF}(x; x_d)$ to the trajectory $x_d(\tau)$, denoted by $D(x)$, mostly remains constant and decreases in some cases:

Proposition 2: Let $x(t)$ be a local solution of $\dot{x} = \mathrm{MVF}(x; x_d)$ for $t \in (-\varepsilon, \varepsilon)$ such that $x(0) = x$. Then, $\dot{D}(x) = 0$ if $x^* \neq x_d(0)$ and $\dot{D}(x) \leq 0$ when $x^* = x_d(0)$. *Proof:* See Appendix A.

Lastly, we note that if x is on the demonstration trajectory, then $\mathrm{MVF}(x_d(\tau)) = \dot{x}_d(\tau)$, meaning that the solution trajectory perfectly recovers the demonstration trajectory.

By utilizing our locally contracting and locally mimicking vector fields, we can define a one-parameter family of local vector fields, which we call a **Local Guidance Vector Field (GVF)**, as follows:

Definition 3: Let $\mathrm{CVF}(x; x_d)$ and $\mathrm{MVF}(x; x_d)$ be locally contracting and mimicking vector fields, a vector field in $\mathcal{U}$ defined as

$$\mathrm{GVF}(x; \eta, x_d) = \mathrm{MVF}(x; x_d) + \eta\, \mathrm{CVF}(x; x_d) \in T_x\mathcal{U}, \tag{19}$$

where $\eta > 0$, is said to be a local guidance vector field to $x_d(\tau)$.

We can control $\eta > 0$ to balance between the mimicking and contracting behaviors. As η increases, a GVF's behavior gradually shifts from mimicking to contracting.

One practical way to select an appropriate η is for the user to provide a small set of supplementary demonstration trajectories, which can then be used to optimize η for the best fit. Fig. 4 illustrates an example where a single additional trajectory is used to tune η.

We show that all local solutions of a GVF are in the direction of reducing $D(x)$.

Proposition 3: Let $x(t)$ be a local solution of $\dot{x} = \mathrm{MVF}(x; x_d) + \eta\, \mathrm{CVF}(x; x_d)$ for $t \in (-\varepsilon, \varepsilon)$ such that $x(0) = x$ for some $\eta > 0$. Then, $\dot{D}(x) \leq -\eta D(x)$, where the equality holds if $x^* \neq x_d(0)$.

Proof: See Appendix A.

As shown in Proposition 3, our GVF is locally exponentially convergent, or *contracting*, toward $x_d(\tau)$ with a rate of η, since the distance $D(x)$ from x to $x_d(\tau)$ satisfies $\dot{D} \leq -\eta D$ (which leads to $D(t) \leq D(0)e^{-\eta t}$ by the comparison lemma).

Remark 1: From a contraction-theoretic perspective, the GVF is *partially contracting*: instead of collapsing to a single point, every trajectory converges exponentially to the target manifold $\{x \in \mathcal{M} \mid D(x) = 0\}$. In a fully contracting system, all virtual displacements satisfy $\|\dot{\delta x}\| \leq -\alpha \|\delta x\|$. Partial contraction relaxes this requirement by demanding the same inequality only for an auxiliary coordinate $z = h(x)$, i.e. $\|\dot{\delta z}\| \leq -\alpha \|\delta z\|$, which guarantees convergence to the invariant manifold $h(x) = z^\star$ [66]. Choosing $z = D(x)$ and applying Proposition 3 yields $|\dot{\delta z}| \leq -\eta |\delta z|$, hence the GVF is partially contracting toward $z = 0$ with rate η, where the attractive manifold would be $D(x) = 0$.

Recall that our goal is to construct a *globally* asymptotically stable dynamical system. However, the local vector fields constructed in the section cannot be well-defined globally because, at points where multiple nearest configurations to $x_d(\tau)$ exist or multiple geodesics exist, both the MVF and CVF can have multiple possible vectors and thus become ill-defined. Even if we choose only one among them, the resulting vector field becomes discontinuous in these regions. Mathematically, this discontinuity makes it challenging to address even the existence of solutions in such cases.

In the subsequent section, we address this issue by constructing a discrete-time dynamical system using the local guidance vector fields. We show that, with minor modifications and under mild assumptions, this system achieves global asymptotic stability. One might wonder whether the stability proof under time discretization is limited, given that the real world evolves in continuous time. However, because real-world control loops operate at a fixed sampling period Δt and are subject to actuation bandwidth limits, our proof in fact reflects the actual closed-loop dynamics, which are inherently discrete rather than continuous.

B. Global Stability Guarantee via Discretization of GVFs

Consider the following discrete-time dynamical system defined by using GVFs:

$$\begin{aligned} v_t &= \mathrm{GVF}(x_t; \eta, x_d) \\ x_{t+1} &= \exp_{x_t}\left(\frac{v_t}{\|v_t\|}\min(\|v_t\|, V_{\max})\Delta t\right) \end{aligned} \tag{20}$$

where $\Delta t > 0$ is a constant time interval and $V_{\max} > \max_\tau \|\dot{x}_d(\tau)\|$ is the clipping value that bounds the maximum speed of the velocity field. In this section, we will show that, for almost all initial configurations $x_0 \in \mathcal{M}$, there exists a sufficiently small step size Δt such that the discrete-time system brings the point x_0 arbitrarily close to the desired trajectory $x_d(\tau)$ and eventually near the terminal point.

We first rule out the set of points that has multiple nearest points to $x_d(\tau)$ or multiple geodesics to its nearest configuration. We assume that this set has measure zero, which is a mild assumption. For instance, consider $x_d(\tau)$ in a 2D space as shown in Fig. 5; the discontinuous region (represented by the green dotted line) forms a one-dimensional space. Let the space excluding this measure-zero set be denoted by $\mathcal{M}_s$, where every $x \in \mathcal{M}_s$ has a unique nearest configuration and geodesic. Furthermore, we assume that starting from an initial point $x_0 \in \mathcal{M}_s$, the solution trajectory $(x_0, x_1, \ldots)$ remains in $\mathcal{M}_s$, avoiding the measure-zero set at every step.

Below, we first show in Proposition 4 that, under some mild additional assumptions, by choosing a sufficiently small Δt, the distance $D(x_t)$ from a particle x_t following (20) to the trajectory $x_d(\tau)$ decreases to within a suitably small range, regardless of the particle's initial point $x_0 \in \mathcal{M}_s$. Furthermore, the particle becomes trapped within a certain distance of $x_d(\tau)$. We then show in Proposition 5 that, with an additional mild assumption, the discrete-time dynamical system (20) is almost globally asymptotically stable at $x_d(T)$.

Proposition 4: Suppose that the minimum distance to $x_d(\tau)$, given by $D : \mathcal{M}_s \to \mathbb{R}$, is differentiable in $\mathcal{M}_s$, and that ∇D is Lipschitz continuous with Lipschitz constant L[2]. In addition, suppose there exists a sufficiently small time step $\Delta t > 0$ and a threshold value $\varepsilon > 0$ such that:

(1) $\|\mathrm{GVF}(x; \eta, x_d)\| \leq V_{\max}$ for all $x \in \mathcal{M}_s : D(x) \leq \varepsilon$.
(2) For every initial configuration $x_0 \in \mathcal{M}_s$, the infinite sequence of a solution trajectory of (20), denoted by $X := (x_0, x_1, \ldots)$, satisfies the following conditions: for almost all configurations $x_t \in X$, with the exception of a finite number of indices t, the entire geodesic connecting x_t and x_{t+1} remains within $\mathcal{M}_s$. In other words, the geodesic does not traverse any discontinuous regions. Moreover, exceptions to this condition occur only in regions where $D(x_t) > \varepsilon$.

Then, for any such ε, there exists a sufficiently large N such that $D(x_t) \leq \varepsilon$ for all $t > N$ if

$$\Delta t < \min\left(\frac{1}{\eta}, \frac{2\eta\varepsilon}{LV_{\max}^2}\right) \tag{21}$$

for all $x_0 \in \mathcal{M}_s$.

Proof: See Appendix A.

The supposition regarding the differentiability of D and the Lipschitz continuity of ∇D is mild, given sufficient smoothness of $x_d(\tau)$. Notably, when the nearest point from x to the trajectory $x_d(\tau)$ lies at the boundary points, D cannot be guaranteed to be more than twice differentiable. Therefore, we rely on a more reasonable assumption: the Lipschitz continuity.

The assumption (1) regarding the existence of ε is very mild, as the maximum speed clipping value $V_{\max}$ is set higher than the maximum speed of $x_d(\tau)$.

The last assumption (2) is, again, mild. To provide an intuitive understanding, Fig. 5 illustrates a scenario where assumption (2) holds. A portion of the discrete-time system's solution trajectory $(x_0, x_1, \ldots, x_t, \ldots)$ from (20) is visualized as red dots. First, note that among the geodesics (red lines), only one passes through the discontinuous region, thereby violating the assumption. We emphasize that such violation

[2]The gradient and Lipschitz continuity are defined within the framework of a Riemannian manifold, with respect to the Riemannian metric.

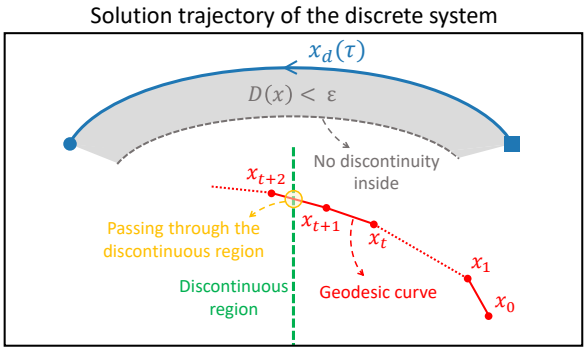

Fig. 5. An illustration of the assumptions in Proposition 4. Given an initial configuration $x_0 \in \mathcal{M}_s$ and corresponding solution trajectory $X = (x_0, x_1, \ldots, x_t, \ldots)$, the geodesics between two adjacent configurations (red lines) generally do not pass through the discontinuous region (green dotted line), except for at time step $t+1$, where the geodesic passes through the discontinuous region, resulting in a finite number of (one) violations of the assumptions in Proposition 4. In region where $D(x) < \varepsilon$ (gray-colored area), there is no discontinuity, thus no violation of the assumptions exist inside.

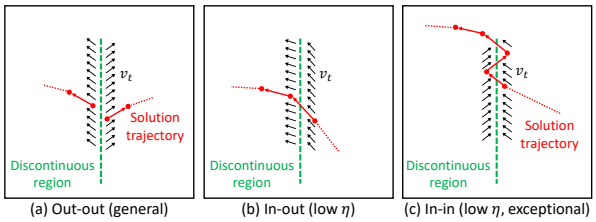

Fig. 6. Three possible scenarios of the discrete-time system in (20) near discontinuous regions. (a) Most general case. Because the contracting velocity draws x_t to the nearest point, x_t does not pass through the discontinuous region, resulting in no violation of the assumptions in Proposition 4. (b) When η is small and the GVF is mostly dominated by the MVF, x_t may enter the discontinuous region from one side and exit from the other side, causing a single violation. (c) In certain extreme low η cases, where the two possible solutions of $\tau^*(x)$ within the discontinuous region are far apart, both mimicking velocities can point toward the discontinuous region (see the rightmost figures in Fig. 9). In this situation, x_t oscillates within the discontinuous region, but eventually escapes the discontinuous region. This possibly leads to more than one, but still finitely many, violations of the assumptions.

cases – where the geodesic between two adjacent configurations passes through the discontinuity region – are very rare and finite in most practical scenarios (see Fig. 6). Second, by choosing a sufficiently small ε, the region $D(x) \leq \varepsilon$ (depicted in gray) can be restricted so that it does not intersect with the discontinuous region (depicted in green). This ensures that exceptions only occur outside this region.

From Proposition 4, we gain valuable insights into the upper bound of Δt. First, note that this upper bound is proportional to ε and inversely proportional to L and $V_{\max}^2$. Intuitively, these relationships are reasonable because: (i) it becomes more challenging to satisfy $D(x_N) < \varepsilon$ as ε decreases, and (ii) for higher curvature of the reference trajectory (L) or greater speed of the system ($V_{\max}^2$), a smaller Δt is required to prevent the accumulation of distance caused by mimicking behavior.

Second, the upper bound exhibits a mixed dependence on η. When η is small, $\min\left(\frac{1}{\eta}, \frac{2\eta\varepsilon}{LV_{\max}^2}\right) = \frac{2\eta\varepsilon}{LV_{\max}^2}$, so the upper bound is proportional to η. Conversely, when η is large, $\min\left(\frac{1}{\eta}, \frac{2\eta\varepsilon}{LV_{\max}^2}\right) = \frac{1}{\eta}$, and the upper bound becomes inversely proportional to η. This behavior can be understood as follows: for small η, the system heavily relies on mimicking the reference trajectory, and increasing η directs the system's velocity more toward the contracting direction, allowing for a larger Δt. However, if η becomes excessively large, the system's overall speed increases significantly, necessitating a smaller Δt to prevent divergence.

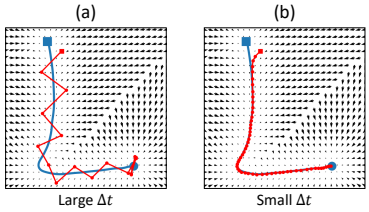

Fig. 7. Comparison of two discrete-time systems with different choices of Δt. The blue curves represent the reference trajectories, while the red polylines and nodes depict the solution trajectories and intermediate states of the discrete-time systems. (a) Too large Δt leads to divergence. (b) Small Δt results in stable convergence to the reference trajectory.

Now, under a very mild additional assumption, we show that if Δt satisfies Proposition 4, for all $x_0 \in \mathcal{M}_s$, x_t converges to the terminal point $x_d(T)$, i.e., the system is globally asymptotically stable at $x_d(T)$.

Proposition 5: Let the assumptions of Proposition 4 hold, and suppose that ε and Δt are chosen to satisfy its conclusions. Further assume that ε is sufficiently small such that, for any x_0, there exists a sufficiently large N satisfying $\tau^*(x_N) = T$ (i.e., the nearest point to x_d eventually reaches the terminal point). Then, the discrete system is globally asymptotically stable, meaning that for all $x_0 \in \mathcal{M}_s$, it holds that $\lim_{t\to\infty} x_t = x_d(T)$.

Proof: See Appendix A.

The assumption in Proposition 5 is in most cases naturally satisfied. By Proposition 4, x_t can be contained within $D(x_t) < \varepsilon$. When ε is sufficiently small, x_t remains in close proximity to $x_d(\tau)$, thus $x_t \approx x_d\big(\tau^*(x_t)\big)$. Consequently, x_t closely follows the reference trajectory until it reaches the final point, satisfying $\tau^*(x_t) = T$. Therefore, it is very reasonable to assume the existence of a sufficiently small ε for which the assumption holds.

It is important to emphasize that Propositions 4 and 5 primarily illustrate that, an appropriate choice of Δt ensures the global asymptotic stability of our discrete-time system. While an upper bound for Δt has been provided, identifying its exact value is often more practical and efficient through experimental tuning. Fig. 7 illustrates two discrete-time systems with large and small values of Δt. In Fig. 7(a), where Δt is large, the state x_t oscillates around the reference trajectory, failing to converge. In contrast, Fig. 7(b) shows that a small Δt enables stable convergence to the reference trajectory.

In practice, there are several challenges to implementation. First, $x_d(\tau)$ is provided as a sequence of configurations, and the nearest point on the trajectory can only be approximated by searching for the minimum within the finite set. A particular issue arises when $x_d(\tau)$ is given as a sparse set of points. In this case, x^* changes discretely, causing the velocity vector to change discontinuously, which can result in a jittery motion. However, increasing the sampling density of x_d to mitigate this issue inevitably exacerbates the computational cost of geodesic and parallel transport calculations, creating an inherent trade-off. Moreover, these computations can be especially expensive when closed-form solutions are unavailable. These numerical

and computational challenges motivate the introduction of a parametric continuous vector field model in the subsequent section to approximate the proposed discrete-time system GVF.

C. Behavior-Controllable Stable Dynamics Models

To address these computational challenges and smooth out the discontinuities caused by discretization, we propose continuous neural network vector field models designed to approximate GVFs, which we refer to as **Behavior-Controllable Stable Dynamics Models (BCSDMs)**. We define the BCSDM as the following weighted sum of two separate vector fields:

$$f_\theta(x,\eta) := m_\theta(x) + \eta\, c_\theta(x), \tag{22}$$

where $c_\theta : \mathcal{M} \to T\mathcal{M}$ and $m_\theta : \mathcal{M} \to T\mathcal{M}$ are continuous parametric vector fields designated to represent contracting and mimicking behaviors, respectively.

Constructing continuous parametric vector field models is particularly challenging when dealing with a manifold $\mathcal{M}$. Care must be taken with both the input and output representations. There are two ways to represent the input configuration $x \in \mathcal{M}$. One is to express it in local coordinates, and the other is to use an implicit representation $x \in \mathbb{R}^n$. In the case of the former, it is difficult to ensure the continuity of the vector field as illustrated in Fig. 1(a), but choosing the latter naturally results in a continuous model.

A slightly more challenging part regards the representation of the output, due to the constraints that the output vector at x must lie in the tangent space $T_x\mathcal{M}$. Here, we let the model output a vector in the ambient space $\mathbb{R}^n$, and then project it onto $T_x\mathcal{M}$ using the projection mapping π defined in (3).

Although a GVF is stable, fitting it with a parametric model may compromise its stability. Nevertheless, we show that if the fitting error is sufficiently small, the stability is largely preserved:

Proposition 6: Let $f : \mathcal{M} \to T\mathcal{M}$ be a continuous vector field, such that $\|f(x) - \mathrm{GVF}(x)\| \le \varepsilon$ for all $x \in \mathcal{M}$. Then, for every initial condition $x(0) \in \mathcal{M}$, there exists t_c such that the corresponding solution trajectory $x(t)$ of $\dot{x} = f(x)$ satisfies $D(x(t)) \le \frac{\varepsilon}{\eta}$ for all $t > t_c$.

Proof: See Appendix A.

Proposition 6 states that all trajectories enter an $\frac{\varepsilon}{\eta}$-tube and remain within it thereafter. This fact provides the intuition that, if the fitting error is sufficiently small, the trajectories closely contract toward the reference trajectory and follow it until the trajectories eventually converge near the equilibrium.

It is reasonable to ask whether discontinuities in the GVF lead to large fitting errors near them, and whether such errors could lead to a loss of stability. However, Proposition 6 implies that if a state can escape those erroneous regions, it will eventually converge. In Section V-A, we also empirically observed that they do not affect the resulting stability in our experiments.

D. Deep Operator Vector Field (DeepOVec)

Suppose we are given multiple demonstration trajectories for various tasks. Since a BCSDM is designed to fit a single trajectory, we may need to train multiple BCSDMs to encode all the given trajectories. However, this becomes increasingly memory-intensive as the number of demonstration trajectories grows. In this section, we expand the BCSDM framework to accommodate multiple trajectories, each corresponding to a different task (e.g., trajectories writing distinct letters), in a memory-efficient way as follows:

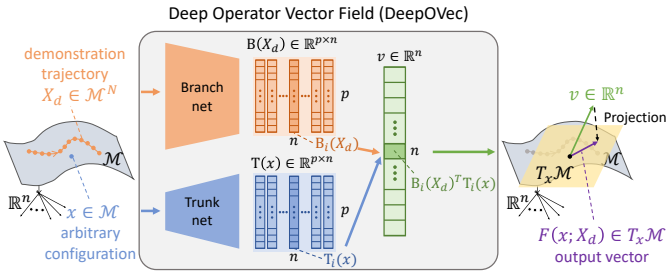

Fig. 8. The illustration of Deep Operator Vector Fields (DeepOVecs). The input of the branch network is a fixed-length configuration sequence $X_d \in \mathcal{M}^N$, and the input of the trunk network is a configuration $x \in \mathcal{M}$. The outputs of the branch network and trunk network form a vector $v \in \mathbb{R}^n$, where the i-th element of v is defined as the inner product of $\mathrm{B}_i(X_d)$ and $\mathrm{T}_i(x)$. v is then mapped to $T_x\mathcal{M}$ through the projection mapping π: $F(x;X_d) = \pi(v)$.

$$f_\theta(x,\eta;x_d) := m_\theta(x;x_d) + \eta\; c_\theta(x;x_d), \tag{23}$$

where f_θ, c_θ, and m_θ now incorporate an additional input, the demonstration trajectory x_d.

We note that c_θ and m_θ map trajectories to vector fields, i.e., they act as operators. Inspired by the Deep Operator Network (DeepONet) framework introduced by Lu et al. [16], which specializes in the memory-efficient learning of operators, we propose a variant aimed at vector fields: **Deep Operator Vector Fields (DeepOVec)**.

We start by providing our problem settings. We consider the demonstration trajectory $x_d(\tau)$ as a fixed-length configuration sequence, i.e., $X_d := (x_d(\tau_1), x_d(\tau_2), \ldots, x_d(\tau_N)) \in \mathcal{M}^N$, where $0 = \tau_1 < \tau_2 < \cdots < \tau_N = T$. We use the same DeepOVec architecture for c_θ and m_θ, and generally denote it by $F : \mathcal{M} \times \mathcal{M}^N \to T\mathcal{M}$ such that $(x, X_d) \mapsto F(x;X_d) \in T_x\mathcal{M}$.

A DeepOVec F consists of a branch network $\mathrm{B} : \mathcal{M}^N \to \mathbb{R}^{p\times n}$ and a trunk network $\mathrm{T} : \mathcal{M} \to \mathbb{R}^{p\times n}$ and is defined as follows:

$$F(x;X_d) := \pi\left(\begin{bmatrix} \mathrm{B}_1(X_d)^T\mathrm{T}_1(x) \\ \vdots \\ \mathrm{B}_n(X_d)^T\mathrm{T}_n(x) \end{bmatrix}\right), \tag{24}$$

where $\mathrm{B}_i(X_d) \in \mathbb{R}^p$ and $\mathrm{T}_i(x) \in \mathbb{R}^p$ are the i-th columns of $p \times n$ matrices $\mathrm{B}(X_d)$ and $\mathrm{T}(x)$, respectively, and $\pi : \mathbb{R}^n \to T_x\mathcal{M}$ is the projection map in (3); see Fig. 8. $\mathrm{T}_i(x)$ is a vector of p basis functions and $\mathrm{B}_i(X_d)$ is a vector of p scalar coefficients. The inner product between them results in a linear basis function model for the i-th vector component. This architecture can approximate any nonlinear continuous operator as p goes to infinity, according to the universal approximation theorem for operators [16].

The DeepOVec has a distinctive feature: significantly enhanced computation and memory efficiency during the inference. This is because we do not need to compute $\mathrm{B}(X_d)$ during the inference nor cache the network parameters of B

and X_d. Note that, to compute the DeepOVec output at x for the task described by X_d, it is sufficient to use pre-computed $\mathrm{B}(X_d)$. This significantly reduces the computation and memory costs. In the remaining sections, we denote a multiple-task version of BCSDM equipped with DeepOVec-based c_θ and m_θ as Behavior-Controllable Deep Operator Vector Fields (BC-DeepOVecs).

V. EXPERIMENTS

In this section, we assess the effectiveness of the Behavior-Controllable Stable Dynamics Models (BCSDM) by conducting comparative experiments against existing state-of-the-art methods. Our evaluations are conducted in two distinct robotic task spaces: a letter-writing task on the surface of the 2-sphere (S^2), requiring mimicking behavior, and an umbrella-inserting task within the robot task space SE(3), demanding contracting behavior. These tasks are selected to highlight the mimicking and contracting properties of BCSDMs.

First, in Section V-A, we provide a stability analysis of continuous BCSDMs in the presence of fitting errors that naturally arise from discontinuities in the guidance vector fields. In Section V-B and Section V-C, we compare BC-DeepOVecs, the multiple-task version of BCSDM equipped with DeepOVecs, to the existing methods in S^2 and SE(3) respectively. Finally in Section V-D, we verify the performance and memory efficiency of the proposed DeepOVec compared to a naive fully connected neural network.

The baseline methods include three Lyapunov function-based methods, two diffeomorphism-based methods, and one contraction theory-based method. For the Lyapunov function-based methods, we consider *Stable Estimator of Dynamical Systems (SEDS)* [2], which use predefined quadratic Lyapunov functions, *Stable Deep Dynamics Models (SDDM)* [10], which rely on neural network Lyapunov functions, and *Locally Active Globally Stable Dynamical Systems (LAGS-DS)*, which introduces an adjustable contraction behavior, via a hyperparameter κ (see Section II-D for more discussion). For the diffeomorphism-based methods, we consider the method introduced by [14], which we refer to as *ManiFlows*, that defines diffeomorphisms in almost global coordinate charts and *Riemannian Stable Dynamical Systems (RSDS)* [15], which utilize dynamic local charts to define diffeomorphisms. For the contraction theory-based method, we consider *Neural Contractive Dynamical Systems (NCDS)* that defines a deep learning-based inherently contractive system. We note that since SEDS, SDDM, and LAGS-DS are originally developed for Euclidean configuration spaces, we define them using almost global coordinate charts on each Riemannian configuration manifold. We mainly report three evaluation metrics in Sections V-B and V-C, with detailed definitions provided in Appendix B. These metrics include the trajectory fitting error, the mimicking error, and the contraction rate. Other metrics (e.g., success rate) are introduced in the corresponding sections.

Training setup Each demonstration trajectory is given as a sequence of configuration–velocity pairs $\{(x_i, \dot{x}_i)\}_{i=1}^{N}$, where N denotes the trajectory length. All models are optimized with an MSE loss on the demonstrations, $\sum_i \|f(x_i) - \dot{x}_i\|^2$. In addition, BC-DeepOVec uses auxiliary losses to fit MVF and CVF, $\sum_j \|m_\theta(x_j) - \mathrm{MVF}(x_j)\|^2 + \|c_\theta(x_j) - \mathrm{CVF}(x_j)\|^2$, where x_j is sampled from the configuration spaces. In each experiment, baseline models were trained separately – one model per trajectory – whereas BC-DeepOVec was trained as a single model jointly on all trajectories. See Appendices C-A and C-B for more details.

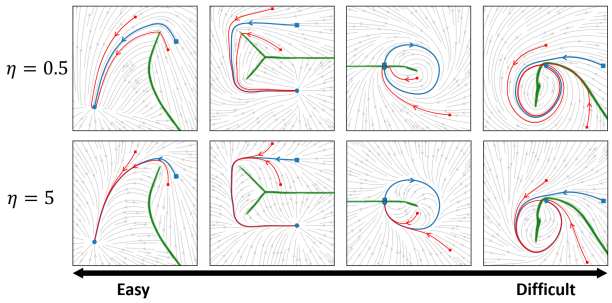

Fig. 9. Illustration of BCSDMs in $\mathbb{R}^2$ learned for various trajectories. Blue curves represent demonstration trajectories, red curves represent sample trajectories, and green areas indicate discontinuous regions of BCSDMs. The first row shows BCSDMs with $\eta = 0.5$, and the second row shows BCSDMs with $\eta = 5$. As we move from left to right, the difficulty of fitting the demonstration trajectories increases as the stable points get closer to discontinuous regions.

A. Stability Study of BCSDM

As mentioned in Section IV-C, fitting discontinuous GVFs with continuous BCSDMs might lead to fitting errors near discontinuities. It is questionable whether the fitting errors could affect the stability, especially when stable points are near discontinuous regions or solution trajectories go through these regions.

In this subsection, we conduct experiments on BCSDMs to verify whether their stability remains intact. We consider four different demonstration trajectories, as shown by the blue curves in Fig. 9. Moving from left to right, we design the discontinuous regions (the green-colored areas) to get closer to the stable point, thereby increasing the impact of fitting errors in these regions on the system's stability. The first row of Fig. 9 shows BCSDMs with $\eta = 0.5$, and the second row shows BCSDMs with $\eta = 5$. The red curves in Fig. 9 visualize some solution trajectories of the systems, all of which successfully converge to the stable points, even when discontinuous regions are very close to the stable points as in the third and fourth columns.

One notable observation in the BCSDM shown in the last column of Fig. 9 is that the red solution trajectory, which starts below the discontinuous regions, initially passes very close to the stable point and then circles back to reach it. If it had followed the GVF, it would have converged directly to the stable point upon initially passing nearby. This indicates that the BCSDM did not perfectly fit the corresponding GVF. Nevertheless, as shown in the figure, the trajectory ultimately converges to the stable point, demonstrating BCSDM's ability to maintain global asymptotic stability even with imperfect fitting. Surprisingly, through sufficiently many trials, we empirically confirm that all solution trajectories converge when starting from randomly sampled configurations.

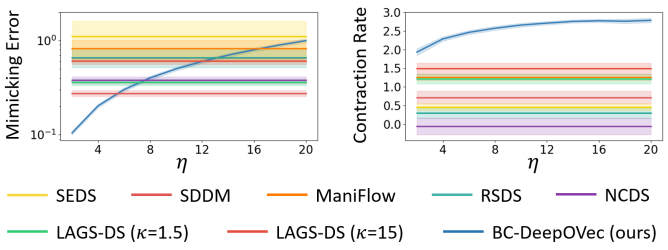

Fig. 10. Mimicking error and contraction rate measures of BC-DeepOVecs with varying η and baseline models.

B. Letter-Writing Task on Spherical Surface

In this section, we consider a letter-writing task on the spherical surface, S^2, embedded in $\mathbb{R}^3$. This task requires the solution trajectories to closely mimic the demonstration trajectories, emphasizing the importance of mimicking behavior over contracting behavior. To generate the dataset, we utilize the LASA dataset [2] and transform its configuration space from the $\mathbb{R}^2$ plane to the spherical surface S^2. The final dataset comprises 26 demonstration trajectories, each corresponding to a different letter. We first conduct a quantitative comparison in terms of fitting error, mimicking error, and contraction rate, followed by a qualitative comparison of each model's performance.

TABLE I
FITTING ERROR (FE); THE LOWER, THE BETTER THE FIT. MIMICKING ERROR (ME); THE LOWER, THE BETTER THE MIMICRY. CONTRACTION RATE (CR); THE HIGHER, THE GREATER THE CONTRACTION.

Method	FE (↓)	**ME (↓)**	CR (↑)
SEDS	0.510	1.109	0.456
SDDM	**0.006**	0.276	0.716
LAGS-DS ($\kappa = 1.5$)	0.339	0.361	1.213
LAGS-DS ($\kappa = 15$)	0.391	0.606	1.490
ManiFlow	0.113	0.825	1.257
RSDS	0.053	0.660	0.295
NCDS	0.118	0.380	-0.062
BC-DeepOVec (ours, $\eta = 2$)	**0.018**	**0.105**	1.971
BC-DeepOVec (ours, $\eta = 20$)	0.088	0.999	**2.537**

Table I presents the three metrics averaged over 26 demonstration trajectories. The fitting errors (FEs) for the deep learning-based methods (SDDM, ManiFlow, RSDS, NCDS, and BC-DeepOVecs) are significantly lower than those of SEDS and LAGS-DS, demonstrating their superior expressiveness. It can be observed that BC-DeepOVec with $\eta = 2$ exhibits the lowest Mimicking Error (ME), while BC-DeepOVec with $\eta = 20$ exhibits the highest Contraction Rate (CR). Since the letter-writing task requires mimicking behavior, the best-performing method is BC-DeepOVec with $\eta = 2$, outperforming all baseline models. Fig. 10 shows the mimicking error (ME) and contraction rate (CR) of the baseline models and BC-DeepOVec with various η. It is clear that both the ME and CR monotonically increase as η increases.

Fig. 11 illustrates the streamlines of the baseline models and BC-DeepOVecs with varying η. The blue lines represent demonstration trajectories, while the red lines indicate some solution trajectories. Firstly, we observe that SEDS and LAGS-DS are unable to fit the 'S'-shaped trajectory (see the orange-colored area) due to the limitations of the GMM-based construction, aligning with the results in Table I. Additionally, the Euclidean methods (SEDS and SDDM) defined in local coordinates result in discontinuous regions (green-colored areas). Although ManiFlow and RSDS successfully fit the given trajectory, they exhibit undesired jerky motions in the yellow-colored area due to overfitting. Finally, in LAGS-DS, lowering κ does not result in *mimicking* behavior. This is because LAGS-DS does not include an explicit mechanism for mimicking.

In contrast to the baseline models, the BC-DeepOVecs successfully fit the demonstration trajectories and are continuous. BC-DeepOVec with $\eta = 20$ exhibits sharply bent solution trajectories. On the other hand, BC-DeepOVec with $\eta = 2$ closely resembles the shape of the given letter 'S', indicating that a low value of η is suitable for the letter-writing task.

It is natural to infer that, since $\eta \approx 0$ would result in the lowest mimicking error, it would likely perform best. However, when η is too low, the solution trajectories tend to follow the demonstration trajectory in parallel until the outputs of MVF become 0, then contract sharply to the stable point. This results in an overall shape that differs from the demonstration trajectory. Therefore, BC-DeepOVec with an adequate η should be chosen.

C. Umbrella-Inserting Task in SE(3)

In this section, we consider an umbrella-inserting task in the task space of robotic manipulators, SE(3). The task involves inserting an umbrella into an assigned hole in the umbrella holder. Since the umbrella must be inserted into a narrow hole, fast contraction is crucial for the task's success. We collect five robot SE(3) trajectories from a human expert performing the umbrella-inserting task (see Fig. 12), with each trajectory corresponding to the insertion of the umbrella into a different hole.

We conduct umbrella-inserting experiments using the 7-DOF Franka Panda robot arm, first in the PyBullet simulator and then in a real-world environment. To convert the end-effector velocities to robot joint velocities, we multiply them by the pseudo-inverse of the robot's body Jacobian matrix [67]. Further details on robot control are in Appendix C-B3.

Experiments in Simulation Environment: Table II presents the fitting error, mimicking error, contraction rate, and success rate, while Fig. 13 illustrates the success rates measured under various disturbances. The disturbances are applied by multiplying a disturbance SE(3) matrix on the right side of the robot end-effector's SE(3) matrix. We pre-assign the scale matrix and the application time of the disturbance, where the scale is defined as the geodesic distance from the identity matrix in SE(3). We categorize the disturbances into two types: (i) disturbances applied at a fixed time (1s) with varying scales, and (ii) disturbances applied at varying times with a fixed scale (0.5). For measuring the success rate (SR) in Table II, we apply disturbances with a scale of 0.5 at 1s.

Table II shows that the trajectory fitting error (FE) is relatively high for most models (except BC-DeepOVec and SDDM), and is especially large for SEDS and LAGS-DS. This suggests that learning SDS in high-dimensional spaces is difficult and that GMM-based models have limited expressiveness. LAGS-DS has a success rate (SR) of zero due to its high FE. In contrast, SEDS – despite its very high FE – achieves higher

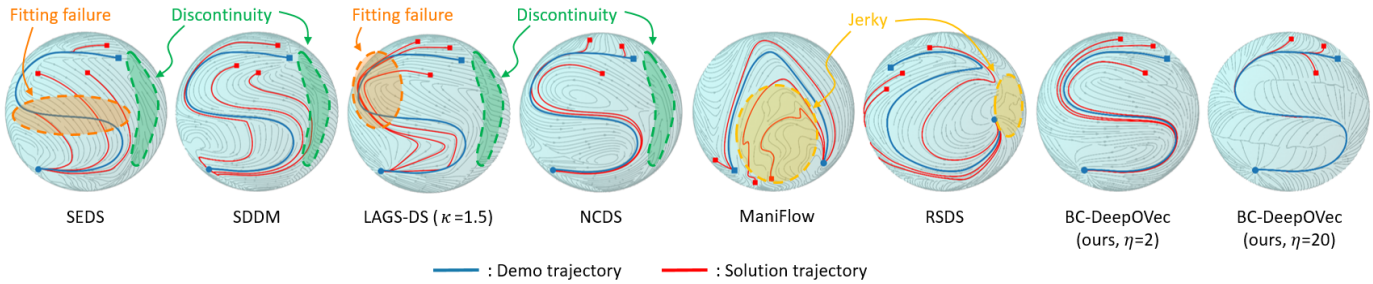

Fig. 11. Streamlines of each SDS learned in S^2. Blue curves and red curves represent demonstration trajectories and sample trajectories, respectively. The green areas are the discontinuous regions and the orange-colored areas show the fitting failures of SEDSs. The areas colored in yellow represent regions where solution trajectories become jerky. For clarity, ManiFlow and RSDS plot dynamical systems corresponding to distinct letters (rather than 'S') to highlight their jerky motions, whereas SEDS, SDDM, and BC-DeepOVec (ours) – which exhibit no such failures – plot only the system for 'S'.

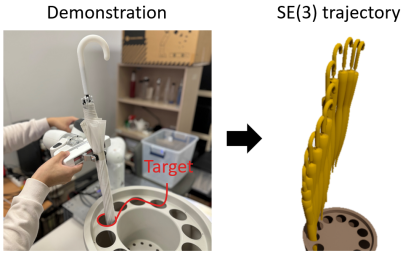

Fig. 12. An illustration of a human demonstration of the umbrella-inserting task. The robot joint angle trajectory is processed to become an SE(3) task space trajectory. The red circle is the assigned target hole.

TABLE II
FITTING ERROR (FE); THE LOWER, THE BETTER THE FIT. MIMICKING ERROR (ME); THE LOWER, THE BETTER THE MIMICRY. CONTRACTION RATE (CR); THE HIGHER, THE GREATER THE CONTRACTION. INFERENCE SPEED (IS) [HZ] AND SUCCESS RATE (SR); THE HIGHER, THE BETTER

Method	FE (↓)	ME (↓)	CR (↑)	IS (↑)	SR (↑)
SEDS	0.757	0.527	0.627	**2688**	0.2
SDDM	0.034	0.110	-0.190	125	0.004
LAGS-DS ($\kappa = 1.5$)	0.856	0.433	0.110	2488	0
LAGS-DS ($\kappa = 15$)	0.912	0.447	0.249	2488	0
ManiFlow	0.161	**0.107**	0.567	95.9	0.116
RSDS	0.202	0.132	0.559	2.75	0.024
NCDS	0.134	0.621	-6.300	14.4	0.001
BC-DeepOVec (ours, $\eta = 0.5$)	**0.031**	0.128	1.151	723	0.416
BC-DeepOVec (ours, $\eta = 5$)	0.045	0.618	**6.558**	723	**0.992**

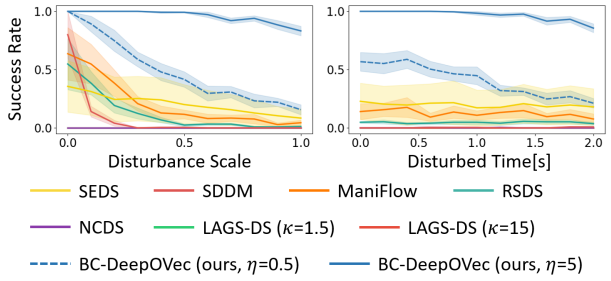

Fig. 13. Success rates measured with (i) varying disturbance scale where the disturbance time is fixed at 1s, and (ii) varying disturbance time where the disturbance scale is fixed at 1.

SR due to the highest CR among baselines, while NCDS's low CR yields an almost zero SR, corroborating CR as the key determinant. Notably, BC-DeepOVec with $\eta = 5$ achieves the highest CR, and BC-DeepOVec with $\eta = 0.5$ achieves the lowest mimicking error (ME). As the umbrella-inserting task demands contracting behavior, BC-DeepOVec with $\eta = 5$ results in the highest SR, outperforming all baseline models by more than double.

In Fig. 13, it is clear that the success rates of BC-DeepOVec with $\eta = 5$ are significantly higher than those of all baseline models in all cases. Furthermore, we observe that the success

TABLE III
SUCCESS RATE (SR) MEASURED WITH NO DISTURBANCE (ND), SMALL DISTURBANCES (SD), OR LARGE DISTURBANCES (LD); THE HIGHER, THE BETTER.

Method	SR (ND)	SR (SD)	SR (LD)
SEDS	2/5	4/10	2/10
SDDM	1/5	0/10	0/10
ManiFlow	1/5	2/10	1/10
RSDS	1/5	2/10	3/10
NCDS	0/5	0/10	0/10
BC-DeepOVec (ours, $\eta = 5$)	**5/5**	**10/10**	**10/10**

rates of SDDM and RSDS drop significantly given any disturbance, indicating that they are overfitted to the demonstration trajectories. Additionally, we observe that the inference speed of RSDS is extremely slow (under 3 Hz). Consequently, although RSDSs adequately fit the demonstration trajectory and are capable of generating successful solution trajectories, they fail in most cases even when there is no additional disturbance.

Fig. 14 illustrates the resulting solution trajectories of the trained models. The yellow umbrellas represent a demonstration trajectory, while the blue umbrellas show trajectories generated by each trained model, starting from the same initial configurations across all models. Baseline models either fail to contract to the demonstration trajectory quickly enough or do not accurately follow it after contracting. In contrast, BC-DeepOVecs with $\eta = 5$ both contract and follow the demonstration trajectory perfectly, while $\eta = 0.5$ is too low to achieve the task effectively.

Experiments in Real-World Environment: We finally execute velocity control using a Franka Panda robot arm in a real-world environment. Unlike the simulation environment, the real-world environment introduces some inherent control errors: (i) errors in controlling the robot to follow the vector field, (ii) imperfect gravity compensation, and (iii) communication delays between the robot control server and the vector field inference server. These errors lead to task failures in baseline models that lack contraction behavior and are not robust to disturbances. Among the compared methods, LAGS-DS was excluded from this experiment on safety grounds, as repeated training attempts in simulation yielded no successful runs and the method exhibited very high trajectory fitting error.

We execute one trial without any disturbance, two trials with small initial disturbances of scale 0.1, and two trials with large initial disturbances of scale 0.3 for each trajectory, resulting in a total of 25 tests per model. We note that the same

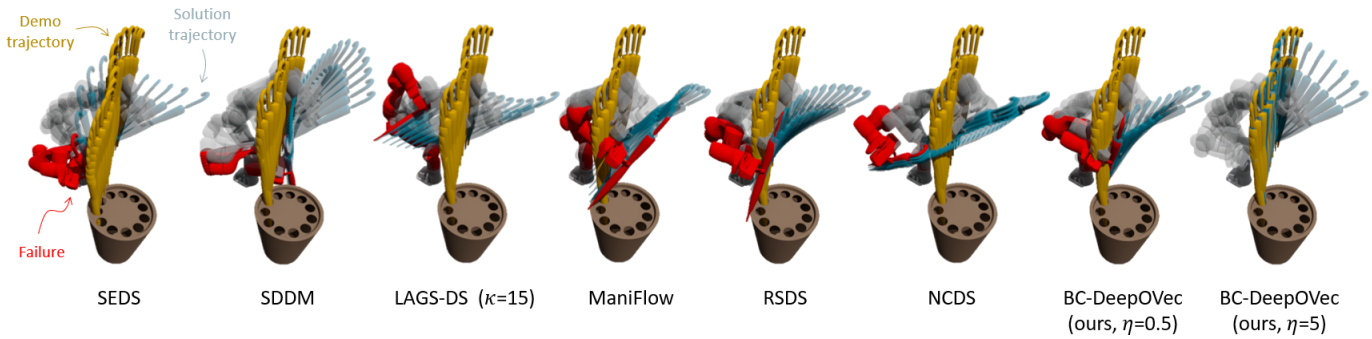

Fig. 14. An illustration of the learned SDS in SE(3). The yellow umbrellas represent the demonstration trajectory, the blue umbrellas represent the solution trajectory starting from the same configuration, and the red-colored robots represent the failures.

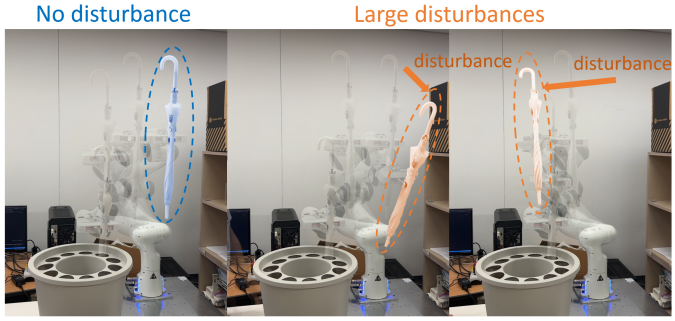

Fig. 15. An illustration of BC-DeepOVec executed in the real-world environment. *Left*: the robot inserting the umbrella without any disturbance. *Middle, right*: the robot inserting the umbrella given large displacements of the robot.

disturbances are used across all models. A trial is considered successful if the umbrella is inserted into the assigned hole at the end.

Table III shows the resulting success rates. BC-DeepOVec's performance is flawless even in the real-world environment, achieving a 100% success rate. This success is largely attributed to the explicit contraction property of BCSDMs, which allows the robot to immediately return to the demonstration trajectory when deviations occur due to control errors. Fig. 15 shows the robot inserting the umbrella using BC-DeepOVec, demonstrating its ability to successfully perform the task even in the presence of significant disturbances. Among baseline models, the deep learning-based baseline models exhibit much lower success rates than in the simulation environment, due to overfitting. Specifically, although SDDM and ManiFlow achieve 100% success rates in the simulation environment without disturbances (see Fig. 13), they fail in almost all trials, demonstrating the sensitivity of deep learning-based models. However, since ManiFlow shows a better contraction rate than SDDM (as shown in Table II), ManiFlow almost succeeds in inserting the umbrella in four more trials, hitting the corner of the assigned hole, while SDDM's failures are far from succeeding. Unlike the deep learning-based methods, SEDS shows the highest success rates among the baseline models due to the high CR.

D. Architecture Study: FCO vs DeepOVec

In this section, we verify the performance and memory efficiency of the proposed DeepOVec framework by comparing it to a naive Fully Connected neural network-based Operator (FCO). An FCO is defined as a fully connected neural network whose inputs are the discretized demonstration trajectory and configuration. We conduct the experiments for the two tasks: the letter-writing task in S^2 and the umbrella-inserting task in SE(3).

TABLE IV
COMPARISON OF DEEPOVEC AND FCO FOR THE LETTER-WRITING TASK IN S^2. CONTRACTING VECTOR FIELD FITTING ERROR (CVF ERROR), MIMICKING VECTOR FIELD FITTING ERROR (MVF ERROR), AND THE NUMBER OF PARAMETERS; THE LOWER, THE BETTER.

Method	CVF Error (↓)	MVF Error (↓)	# Param (↓)
FCO	1.02×10^{-1}	2.15×10^{-1}	17M
DeepOVec	$\mathbf{2.95 \times 10^{-2}}$	$\mathbf{4.95 \times 10^{-2}}$	**11M**

TABLE V
COMPARISON OF DEEPOVEC AND FCO FOR THE UMBRELLA-INSERTING TASK IN SE(3). CONTRACTING VECTOR FIELD FITTING ERROR (CVF ERROR), MIMICKING VECTOR FIELD FITTING ERROR (MVF ERROR), AND THE NUMBER OF PARAMETERS; THE LOWER, THE BETTER.

Method	CVF Error (↓)	MVF Error (↓)	# Param (↓)
FCO	6.44×10^{-2}	3.06×10^{-2}	23M
DeepOVec	$\mathbf{2.95 \times 10^{-3}}$	$\mathbf{8.68 \times 10^{-3}}$	**12M**

Table IV and Table V present the Contracting Vector Field fitting Error (CVF Error), the Mimicking Vector Field fitting Error (MVF Error), and the number of parameters for the two tasks, respectively. In Table IV, it is evident that DeepOVec achieves better fitting for both CVFs and MVFs while using a much smaller number of parameters compared to FCO. This tendency is even clearer in Table V, where DeepOVec's CVF Error is more than 20 times smaller than FCO's, using approximately half the parameters.

VI. CONCLUSION

In this paper, we have proposed the Behavior-Controllable Stable Dynamics Models (BCSDMs), a novel approach for learning Stable Dynamical Systems (SDS) on Riemannian configuration manifolds. BCSDMs uniquely balance contracting and mimicking behaviors, allowing adaptation to a wide range of tasks. We have further extended our framework to a multi-task version by proposing Deep Operator Vector Fields (DeepOVecs), enabling memory-efficient encoding of multiple tasks without sacrificing performance. Through extensive experiments, we have demonstrated the superiority of BCSDM over existing state-of-the-art methods across various metrics.

There are several ways to further enhance our framework. First, the current architecture of the branch network is a fully connected neural network that takes discretized trajectories

as input. This implementation leads to a limitation that the lengths of the trajectory data must all be the same, requiring post-processing. Using neural network models specialized for variable-length time-series or sequence data, such as recurrent neural networks and transformers, could potentially broaden BCSDM's applicability.

Secondly, this paper focuses on constructing DeepOVec, which memorizes and encodes given multiple demonstration trajectories and their corresponding BCSDMs. As a result, if unseen trajectories are provided as inputs to DeepOVec, it may produce meaningless velocity vector outputs. Extending DeepOVec to generalize to unseen trajectories would be an interesting direction for future work. If achieved, DeepOVec could be integrated with existing movement primitives designed to generate desired trajectories, converting trajectory representations into vector field representations that enable robust control against external perturbations.

APPENDIX A
PROOFS OF PROPOSITIONS

In this section, we provide proofs of Propositions 1, 2, 3, 4, 5, and 6.

We first introduce some properties that are used in the proofs. Let $\gamma(s)$ be any geodesic curve. Then $\pi(\gamma''(s)) = 0$, and thus $\langle \gamma''(s), v\rangle = 0$ for any $v \in T_{\gamma(s)}\mathcal{M}$. The geodesic is an auto-parallel curve, and thus $\Gamma^{\gamma(b)}_{\gamma(a)}\gamma'(a) = \gamma'(b)$ for any $a, b \in [0,1]$. The inner product is conserved under the parallel transport, i.e., $\langle v, w\rangle = \langle \Gamma^{y}_{x} v, \Gamma^{y}_{x} w\rangle$ for all $x, y \in \mathcal{M}$ and $v, w \in T_x\mathcal{M}$.

Throughout, we denote the energy of a geodesic curve γ_x by $E(x) := \int_0^1 \langle \gamma_x', \gamma_x'\rangle ds = D(x)^2$. We first introduce three lemmas that are used later in proving the Propositions.

Lemma 1: Let $x_d : [0,T] \to \mathcal{M}$ be a smooth curve in $\mathcal{M}$. Let U be an open subset of $\mathcal{M}$ such that, for any $x \in U$, there exists a unique minimizer of $\min_{\tau\in[0,T]} d^2(x, x_d(\tau))$ – where $d : \mathcal{M}\times\mathcal{M} \to \mathbb{R}$ is the geodesic distance function – denoted by $\tau^*(x)$. Then, $\tau^*(x)$ is continuous in x over U.

Proof: Set $f(x,\tau) := d^2(x, x_d(\tau))$. Since d is continuous and x_d is smooth, f is jointly continuous in (x,τ). Fix $x \in U$ and a sequence $x_n \to x$; put $\tau_n := \tau^*(x_n) \in [0,T]$. By compactness of $[0,T]$, some subsequence $\tau_{n_k} \to \tau_0 \in [0,T]$. For any fixed $\tau \in [0,T]$, optimality gives $f(x_{n_k}, \tau_{n_k}) \le f(x_{n_k}, \tau)$; by joint continuity (i.e., $x_{n_k} \to x$ and $\tau_{n_k} \to \tau_0$ imply $f(x_{n_k}, \tau_{n_k}) \to f(x, \tau_0)$ while $f(x_{n_k}, \tau) \to f(x, \tau)$), letting $k \to \infty$ yields $f(x,\tau_0) \le f(x,\tau)$ for all τ. Thus τ_0 is a minimizer at x, and by uniqueness $\tau_0 = \tau^*(x)$. Hence every convergent subsequence of (τ_n) has the same limit $\tau^*(x)$; by the sequential criterion, $\tau_n \to \tau^*(x)$, so τ^* is continuous on U. ■

Lemma 2: Let $f : \mathcal{M} \to \mathbb{R}$ be a continuous function and $g : \mathcal{M} \to \mathcal{M}$ be a map. For all x such that $f(x) > 0$, x is mapped to the same point y in $\mathcal{M}$, i.e., $g(x) = y$ for all $x : f(x) > 0$. Then, for all $x \in \mathcal{M}$ with $f(x) > 0$, g is differentiable at x and its differential at x is the zero map, i.e., $d_x g(v) = 0$ for all $v \in T_x\mathcal{M}$.

Proof: Let x be a point in $\mathcal{M}$ such that $f(x) > 0$. For any $\varepsilon > 0$, the set $B_\varepsilon(x) = \{x' \in \mathcal{M} \mid |f(x') - f(x)| < \varepsilon\}$ is an open neighborhood around x (due to the continuity of f). By selecting $\varepsilon < |f(x)|$, we can ensure that for all $x' \in B_\varepsilon(x)$, $f(x') > 0$ (since $f(x') > f(x) - \varepsilon$ and $f(x) > 0$). Since $f(x') > 0$ implies $g(x') = y$, g is a constant map in $B_\varepsilon(x)$, i.e., $g(x') = y$ for all $x' \in B_\varepsilon(x)$. Therefore, g is differentiable and the differential $d_x g = 0$. ■

Lemma 3: Let $x(t) \in \mathcal{M}$ be any smooth curve that passes x at $t = 0$, $x^*(x(t))$ be the nearest point from $x(t)$ to the trajectory $\{x_d(\tau) : \tau \in [0,T]\}$ (assuming it is unique), and $\gamma_{x(t)} : [0,1] \to \mathcal{M}$ be the geodesic from $x(t)$ to $x^*(x(t))$ such that $\gamma_{x(t)}(0) = x(t)$ and $\gamma_{x(t)}(1) = x^*(x(t))$. Then $\langle \frac{d}{dt}|_{t=0} x^*(x(t)), \gamma_x'(1)\rangle = 0$.

Proof: Applying the integration by parts formula for the energy of γ_x, we obtain $E(x) = \langle x^*, \gamma_x'(1)\rangle - \langle x, \gamma_x'(0)\rangle - \int_0^1 \langle \gamma_x, \gamma_x''\rangle ds$. Let $x^* = x_d(\tau^*)$; then we may write the energy as a function of τ^* and x as follows

$$E(\tau^*; x) = \langle x_d(\tau^*), \gamma_x'(1)\rangle - \langle x, \gamma_x'(0)\rangle - \int_0^1 \langle \gamma_x, \gamma_x''\rangle ds.$$

Then, we can notice that, given a fixed configuration $x \in \mathcal{M}$, τ^* is the minimizer of the function

$$E(\tau) := \langle x_d(\tau), \gamma_x'(1)\rangle - \langle x, \gamma_x'(0)\rangle - \int_0^1 \langle \gamma_x, \gamma_x''\rangle ds$$

such that $\tau \in [0,T]$. Therefore, we can derive the KKT (Karush–Kuhn–Tucker) conditions with the Lagrangian function $\mathcal{L}(\tau, \mu_1, \mu_2) = E(\tau) + \mu_1(-\tau) + \mu_2(\tau - T)$ with the Lagrange multipliers μ_1, μ_2:

- $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle - \mu_1 + \mu_2 = 0$;
- $\mu_1 \tau^* = 0$, $\mu_1 \ge 0$, $\tau^* \ge 0$;
- $\mu_2(T - \tau^*) = 0$, $\mu_2 \ge 0$, $T - \tau^* \ge 0$.

These conditions lead to the following conclusions:

(A) If $\tau^* = 0$, then $\mu_2 = 0$, which implies $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle = \mu_1 \ge 0$;
(B) If $\tau^* \in (0,T)$, then $\mu_1 = \mu_2 = 0$, which implies $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle = 0$;
(C) If $\tau^* = T$, note that $\dot{x}_d(T) = 0$ (due to our assumption of a zero terminal velocity curve). Hence, $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle = 0$.

Remark 2: If $\langle \dot{x}_d(\tau^*(x)), \gamma_x'(1)\rangle > 0$ for some $x \in \mathcal{M}$, then $\tau^*(x) = 0$ and $x^*(x) = x_d(0)$.

Now, we are ready to show that $\langle \frac{d}{dt}|_{t=0} x^*(x(t)), \gamma_x'(1)\rangle = 0$. Let $x^*(x(0)) = x^*(x) = x_d(\tau^*)$ for some $\tau^* \in [0,T]$, then it is evident that $\frac{d}{dt}|_{t=0} x^*(x(t)) = c\dot{x}_d(\tau^*)$ for some $c \in \mathbb{R}$. Thus, we can write the left-hand side as $\langle c\dot{x}_d(\tau^*), \gamma_x'(1)\rangle$.

The proof proceeds by considering two cases: (i) $\tau^* \in (0,T]$ and (ii) $\tau^* = 0$. In the first case, from the above conclusions (B) and (C), $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle = 0$, which leads to $\langle \frac{d}{dt}|_{t=0} x^*(x(t)), \gamma_x'(1)\rangle = 0$.

In the second case, $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle \ge 0$. If this expression equals zero, the statement holds for the same reason as in the first case. If it is strictly greater than zero, a different argument is required: we will instead show that $\frac{d}{dt}|_{t=0}\, x^*(x(t)) = 0$. First, we note that Lemma 1 states that $\tau^*(x)$ is continuous in x. Then, $\dot{x}_d(\tau^*(x))$ and $\gamma_x'(1)$ inherit the continuity, resulting in continuity of $\langle \dot{x}_d(\tau^*(x)), \gamma_x'(1)\rangle$ with respect to x. By substituting $f(x)$ with $\langle \dot{x}_d(\tau^*(x)), \gamma_x'(1)\rangle$ and $g(x)$ with $x^*(x)$ in Lemma 2, thanks to Remark 2, $d_x(x^*) = 0$ for x satisfying $\langle \dot{x}_d(\tau^*), \gamma_x'(1)\rangle > 0$, which trivially leads to $\frac{d}{dt}|_{t=0} x^*(x(t)) = 0$. ■

Now, with the above three lemmas, we will prove the Propositions. Consider a smooth trajectory $x(t)$ and the energy function evaluated at $x(t)$: $E(x(t)) := \int_0^1 \langle \gamma'_{x(t)}, \gamma'_{x(t)} \rangle ds$. Then, the time derivative of the energy function is $\frac{dE(x)}{dt} := 2\int_0^1 \langle \gamma'_x, \gamma'_x \rangle ds$. We use the integration by parts and get $\frac{dE(x)}{dt} = 2[\langle \dot{\gamma}_x, \gamma'_x \rangle]_0^1 - 2\int_0^1 \langle \dot{\gamma}_x, \gamma''_x \rangle ds = 2\langle \gamma'_x(1), \dot{\gamma}_x(1) \rangle - 2\langle \gamma'_x(0), \dot{\gamma}_x(0) \rangle$ (since $\langle \dot{\gamma}_x, \gamma''_x \rangle = 0$). We note that x^* is a function of x, denoted by $x^*(x)$, and the time derivative of $E(x)$ can be written as follows:

$$\frac{dE(x)}{dt} = 2\langle \gamma'_x(1), \frac{d}{dt} x^*(x(t)) \rangle - 2\langle \gamma'_x(0), \frac{d}{dt} x(t) \rangle. \tag{25}$$

The first term in (25) vanishes due to the Lemma 3. Therefore, the time derivative (25) is simplified as follows:

$$\frac{dE(x)}{dt} = -2\langle \gamma'_x(0), \dot{x} \rangle, \tag{26}$$

which will be used for the following proofs.

Proof of Proposition 1: $\dot{E} = -2\langle \gamma'_x(0), \gamma'_x(0) \rangle$. Note $\|\gamma'_x(0)\| = \mathrm{Len}(\gamma_x) = D$ (see the preliminary Section III-B). Thus, $\dot{E} = 2D\dot{D} = -2D^2$, implying that $\dot{D} = -D \leq 0$, where the equality holds at $D = 0$ (i.e., x is on $x_d(\tau)$). ■

Proof of Proposition 2: Since $\dot{x} = \Gamma^x_{x^*} \dot{x}^*$, $\gamma'_x(0) = \Gamma^x_{x^*} \gamma'_x(1)$, and the parallel transport preserves the inner product, we obtain $\dot{E} = -2\langle \Gamma^x_{x^*} \gamma'_x(1), \Gamma^x_{x^*} \dot{x}^* \rangle = -2\langle \gamma'_x(1), \dot{x}^* \rangle$. Therefore, $\dot{E} = 2D\dot{D} = 0$ if $\tau^*(x) \in (0, T]$ and $\dot{E} = 2D\dot{D} \leq 0$ if $\tau^*(x) = 0$ (see (A), (B), and (C) in the proof of Lemma 3). ■

Proof of Proposition 3: $\dot{E} = -2\langle \gamma'_x(0), \mathrm{MVF}(x) + \eta \mathrm{CVF}(x) \rangle = -2\langle \gamma'_x(1), \dot{x}^* \rangle - 2\eta \langle \gamma'_x(0), \gamma'_x(0) \rangle = -2\langle \gamma'_x(1), \dot{x}^* \rangle - 2\eta D^2$. Since $\dot{E} = 2D\dot{D}$, $\dot{D} = -\eta D \leq 0$ if $\tau^*(x) \in (0, T]$ and $\dot{D} \leq -\eta D \leq 0$ if $\tau^*(x) = 0$ (see (A), (B), and (C) in the proof of Lemma 3). ■

Proof of Proposition 4: Before beginning the proof, we provide three lemmas that will be used later, and the definition of Lipschitz continuity of vector fields on Riemannian manifolds.

Lemma 4: Let $X = (x_0, x_1, \ldots)$ be an infinite configuration sequence where $x_t \in \mathcal{M}$ for $t = 0, 1, \ldots$, and let $f : \mathcal{M} \to \mathbb{R}$ be a function. Suppose, for any $\varepsilon > 0$, there exists $\alpha > 0$ such that $f(x_{t+1}) - f(x_t) \leq -\alpha$ for all $x_t \in X$ satisfying $f(x_t) > \varepsilon$, except for a finite number of time indices. Then, there exists an integer $N \geq 0$ such that $f(x_N) < \varepsilon$.

Proof: Suppose that there exist N_v points that violate the condition $f(x_{t+1}) - f(x_t) \leq -\alpha$, i.e., $f(x_{t+1}) - f(x_t) > -\alpha$ for $t \in \{k_i\}_{i=1}^{N_v}$. Also, denote $\max_i (f(x_{k_i+1}) - f(x_{k_i})) = M$.

Now, assume that x_t never reaches $f(x_t) < \varepsilon$, thus $f(x_{t+1}) - f(x_t) \leq -\alpha$ for all x_t except for the N_v violations. Then, for a large enough integer $t' \geq \frac{f(x_0) + N_v M}{\alpha} + N_v$, the following holds:

$$\begin{aligned} f(x_{t'}) &= f(x_0) + \sum_{t=0}^{t'-1} (f(x_{t+1}) - f(x_t)) \\ &\leq f(x_0) + (N_v)M - (t' - N_v)\alpha \\ &\leq f(x_0) + (N_v)M - (f(x_0) + N_v M) = 0. \end{aligned} \tag{27}$$

$f(x_{t'}) \leq 0 < \varepsilon$, meaning that the assumption that x_t never reaches $f(x_t) < \varepsilon$ is false. Therefore, there must exist an integer N such that $f(x_N) < \varepsilon$. ■

Lemma 5 (Descent Lemma): We start with the definition of Lipschitz continuity on Riemannian manifolds.

Definition 4 (Lipschitz Continuity of Vector Fields): Let $f : \mathcal{U} \to T\mathcal{U}$ be a continuous vector field in $\mathcal{U}$. The function f is said to be Lipschitz continuous in $\mathcal{U}$ with Lipschitz constant $L > 0$ if

$$\|f(x) - \Gamma^x_y f(y)\| \leq L \, \mathrm{dist}(x, y) \tag{28}$$

for all $x, y \in \mathcal{U}$ and a constant L.

Let $f : \mathcal{M}_s \to \mathbb{R}$ be a differentiable function, and suppose that the Riemannian gradient ∇f is Lipschitz continuous with Lipschitz constant $L > 0$ (See Definition 4). Then, given a geodesic curve $\gamma : [0, 1] \to \mathcal{M}_s$ with $\gamma(0) = x$ and $\gamma(1) = y$,

$$f(y) - f(x) \leq d_x f(\gamma'(0)) + \frac{L}{2} \|\gamma'(0)\|^2. \tag{29}$$

Proof: We write $f(y) - f(x)$ as follows:

$$\begin{aligned} f(y) - f(x) &= \int_0^1 \frac{d}{ds} f(\gamma(s))\Big|_{s=s} ds \\ &= \frac{d}{ds} f(\gamma(s))\Big|_{s=0} + \int_0^1 \Big(\frac{d}{ds} f(\gamma(s))\Big|_{s=s} - \frac{d}{ds} f(\gamma(s))\Big|_{s=0} \Big) ds. \end{aligned} \tag{30}$$

The first term of the right-hand side becomes $\frac{d}{ds} f(\gamma(s))\Big|_{s=0} = d_x f(\gamma'(0))$. The integrand in the second term can be expressed as follows:

$$\frac{d}{ds} f(\gamma(s))\Big|_{s=s} - \frac{d}{ds} f(\gamma(s))\Big|_{s=0} = \langle \nabla f(\gamma(s)), \gamma'(s) \rangle - \langle \nabla f(\gamma(0)), \gamma'(0) \rangle. \tag{31}$$

Since parallel transport preserves the inner product, the right-hand side of (31) can be derived as follows:

$$\begin{aligned} \text{RHS of (31)} &= \langle \nabla f(\gamma(s)) - \Gamma^{\gamma(s)}_{\gamma(0)} \nabla f(\gamma(0)), \ \gamma'(s) \rangle \\ &\leq \|\nabla f(\gamma(s)) - \Gamma^{\gamma(s)}_{\gamma(0)} \nabla f(\gamma(0))\| \|\gamma'(s)\|. \end{aligned} \tag{32}$$

Since ∇f is Lipschitz continuous, $\|f(\gamma(s)) - \Gamma^{\gamma(s)}_{\gamma(0)} \nabla f(\gamma(0))\| \leq L \mathrm{dist}(\gamma(0), \gamma(s)) = Ls\|\gamma'(0)\|$. Then, combining this to (32) leads to the following inequality:

$$\text{RHS of (31)} \leq Ls\|\gamma'(0)\|^2, \tag{33}$$

because $\|\gamma'(0)\| = \|\gamma'(s)\|$ for all s.

Putting this into the integral,

$$\begin{aligned} \int_0^1 \Big(\frac{d}{ds} f(\gamma(s))\Big|_{s=s} - \frac{d}{ds} f(\gamma(s))\Big|_{s=0} \Big) ds &\leq \int_0^1 Ls\|\gamma'(0)\|^2 ds \\ &= \frac{L}{2} \|\gamma'(0)\|^2. \end{aligned} \tag{34}$$

Then, substituting (34) into (30) leads to the inequality of (29). ■

Lemma 6: Let $x_t \in \mathcal{M}_s$ and $D(x_t)$ be a minimum geodesic distance from x_t to $x_d(\tau)$. Suppose that $D(x_t)$ is finite and $D(x_t) > \varepsilon$. Then,

$$-\eta D(x_t) \frac{\min(\|\mathrm{GVF}(x_t; \eta, x_d)\|, V_{\max})}{\|\mathrm{GVF}(x_t; \eta, x_d)\|} \leq -\eta\varepsilon.$$

Proof: Let $\gamma : [0, 1] \to \mathcal{M}_s$ be the arc-length–parametrized minimum geodesic from $x_t = \gamma(0)$ to the nearest point $\gamma(1) = x_d(\tau^*(x_t))$, and set

$$\delta(s) := D(\gamma(s)) = (1 - s) D(x_t) \qquad (0 \leq s \leq 1).$$

Define $c(s) := \mathrm{CVF}(\gamma(s)) = -\delta(s)\nabla D(\gamma(s))$ so that $\|c(s)\| = \delta(s)$, and $m(s) := \mathrm{MVF}(\gamma(s)) = \Gamma_{\gamma(1)}^{\gamma(s)} \dot{x}_d(\tau^*(x_t))$ so that $\|m(s)\|$ is constant in s by parallel transport. Since γ is a geodesic, $\gamma'(s)$ is parallel transported along γ, and $m(s)$ is also parallel transported. Hence $\langle \gamma'(s), m(s)\rangle$ is constant in s and, by the KKT conclusion (A) used in Lemma 3,

$$C \;:=\; \langle \gamma'(s), m(s)\rangle \equiv \langle \gamma'(1), \dot{x}_d(\tau^*(x_t))\rangle \;\geq 0.$$

Since the tail segment of γ from $\gamma(s)$ to $\gamma(1)$ is the minimum geodesic of length $\delta(s)$, we have $c(s) = \frac{\delta(s)}{D(x_t)}\gamma'_{x_t}(s)$. Noting $\langle c(s), m(s)\rangle = \frac{\delta(s)}{D(x_t)} C$, we obtain for $\mathrm{GVF}(\gamma(s)) = \eta\, c(s) + m(s)$ that

$$\|\mathrm{GVF}(\gamma(s))\|^2 = \underbrace{\eta^2}_{a}\,\delta(s)^2 + \underbrace{\frac{2\eta C}{D(x_t)}}_{b}\,\delta(s) + \underbrace{\|m(s)\|^2}_{c} =: \phi(\delta(s)),$$

where $a > 0$, $b \geq 0$, $c \geq 0$. Thus $\phi'(\delta) = 2a\delta + b \geq 0$, so $\sqrt{\phi(\delta)}$ is increasing in δ. Since $\delta(s)$ decreases linearly in s, $\|\mathrm{GVF}(\gamma(s))\|$ decreases in s; in particular, the equation $\|\mathrm{GVF}\| = V_{\max}$ can hold for at most one s, and the two formulas below glue continuously there.

Now analyze $g(\gamma(s)) = -\eta\,\delta(s)\,\frac{\min(\|\mathrm{GVF}(\gamma(s))\|, V_{\max})}{\|\mathrm{GVF}(\gamma(s))\|}$ in two cases.

(i) Unsaturated region $\big(\|\mathrm{GVF}\| \leq V_{\max}\big)$: then $g(\gamma(s)) = -\eta\,\delta(s)$, which is strictly increasing in s because $\delta(s)$ decreases.

(ii) Saturated region $\big(\|\mathrm{GVF}\| > V_{\max}\big)$: write

$$g(\gamma(s)) = -\eta V_{\max}\, r(\delta(s)), \qquad r(\delta) := \frac{\delta}{\sqrt{\phi(\delta)}}.$$

A direct one-line computation gives

$$r'(\delta) = \frac{\phi(\delta) - \frac{\delta}{2}\phi'(\delta)}{\phi(\delta)^{3/2}} = \frac{c + \frac{b}{2}\delta}{\phi(\delta)^{3/2}} \;\geq\; 0.$$

Hence $r(\delta)$ increases with δ, so it decreases with s; multiplying by the negative constant $-\eta V_{\max}$ shows $g(\gamma(s))$ is increasing in s here as well.

Therefore $g(\gamma(s))$ is continuous on $[0,1]$ and (piecewise) increasing, so for any $s \in [0,1]$, $g(\gamma(0)) \leq g(\gamma(s))$. Choose $s_\varepsilon := 1 - \varepsilon/D(x_t)$ so that $\delta(s_\varepsilon) = \varepsilon$. By the stated assumption for the ε-tube (no clipping when $D \leq \varepsilon$), the point $\gamma(s_\varepsilon)$ is in case (i), hence $g(\gamma(s_\varepsilon)) = -\eta\,\varepsilon$. Thus

$$g(x_t) = g(\gamma(0)) \;\leq\; g(\gamma(s_\varepsilon)) = -\eta\,\varepsilon,$$

which is exactly the desired bound. ∎

Now, with the above three lemmas, our proof proceeds in two main steps:

1) We establish an upper bound for Δt such that, for any $x_0 \in \mathcal{M}_s$, there exists an integer $N > 0$ with $D(x_N) < \varepsilon$.
2) We establish a further condition on Δt that guarantees $D(x_t) < \varepsilon$ for all $t > N$.

Step 1. For any t with $D(x_t) > \varepsilon$ such that the geodesic from x_t to x_{t+1} lies in $\mathcal{M}_s$, Lemma 5 and the L–Lipschitz continuity of ∇D give

$$D(x_{t+1}) - D(x_t) \;\leq\; d_{x_t} D\Big(v_t^{\mathrm{clip}}\Big)\,\Delta t \;+\; \frac{L}{2}\,\|v_t^{\mathrm{clip}}\|^2\,\Delta t^2.$$

By Lemma 6, $d_{x_t} D\Big(v_t^{\mathrm{clip}}\Big) = -\eta D(x_t)\,\frac{\min(\|v_t\|, V_{\max})}{\|v_t\|} \leq -\eta\varepsilon$, and by construction $\|v_t^{\mathrm{clip}}\| \leq V_{\max}$. Hence

$$D(x_{t+1}) - D(x_t) \;\leq\; -\eta\varepsilon\,\Delta t \;+\; \frac{L}{2} V_{\max}^2\,\Delta t^2.$$

If $\Delta t < \frac{2\eta\varepsilon}{L V_{\max}^2}$ the right–hand side is $\leq -\alpha$ for some $\alpha > 0$, so $D(x_t)$ strictly decreases at such steps. Assumption (2) allows only finitely many violations and only when $D(x_t) > \varepsilon$; thus, by Lemma 4, there exists N with $D(x_N) < \varepsilon$.

Step 2. For all $t \geq N$ with $D(x_t) < \varepsilon$ and the geodesic segment inside $\mathcal{M}_s$, using (1) and the same bound,

$$D(x_{t+1}) \leq D(x_t) - \eta\,\Delta t\, D(x_t) + \frac{L}{2} V_{\max}^2\,\Delta t^2 = (1 - \eta\Delta t)\, D(x_t) + \frac{L}{2} V_{\max}^2\,\Delta t^2.$$

If $\Delta t \leq \frac{1}{\eta}$, the right–hand side is increasing in $D(x_t) \in [0, \varepsilon]$, so its maximum over that interval occurs at $D(x_t) = \varepsilon$. Therefore

$$D(x_{t+1}) \;\leq\; \varepsilon \;-\; \eta\varepsilon\,\Delta t \;+\; \frac{L}{2} V_{\max}^2\,\Delta t^2 \;\leq\; \varepsilon$$

whenever $\Delta t \leq \frac{2\eta\varepsilon}{L V_{\max}^2}$. Combining the two steps, if

$$\Delta t \;<\; \min\Big(\tfrac{1}{\eta},\; \tfrac{2\eta\varepsilon}{L V_{\max}^2}\Big),$$

then $D(x_t) \leq \varepsilon$ for all $t > N$, as claimed. ∎

Proof of Proposition 5: For $x_0 \in \mathcal{M}_s$, let t be a sufficiently large integer such that $\tau^*(x_t) = T$ and $D(x_t) < \varepsilon$ (such a t exists under the given assumptions). We will show that x_t exponentially converges to $x_d(T)$, which implies global asymptotic stability.

We note that, since $\dot{x}_d(T) = 0$, $\mathrm{MVF}(x_t) = 0$. Thus, $v_t = \eta\mathrm{CVF}(x_t)$ where $\mathrm{CVF}(x_t) = -\gamma'_{x_t}(0)$. Consider

$$x_{t+1} = \exp_{x_t}(v_t\Delta t) = \exp_{x_t}(-\eta\gamma'_{x_t}(0)\Delta t).$$

Since $\exp_{x_t}(-\eta\gamma'_{x_t}(0)\Delta t) = \gamma_{x_t}(\eta\Delta t)$, we get the following equalities

$$\begin{aligned} D(x_{t+1}) &= D\big(\gamma_{x_t}(\eta\Delta t)\big) \\ &= \mathrm{dist}(\gamma_{x_t}(1), \gamma_{x_t}(\eta\Delta t)) \\ &= |1 - \eta\Delta t|\; D(x_t). \end{aligned} \tag{35}$$

Since $\Delta t < \frac{1}{\eta}$, we have $D(x_{t+1}) = (1 - \eta\Delta t)D(x_t)$. Furthermore, the nearest point remains the same, i.e., $\tau^*(x_{t+1}) = T$, because x_t moves to x_{t+1} in the direction that most steeply decreases the geodesic distance. Therefore, $\lim_{t'\to\infty} D(x_{t'}) = \lim_{n\to\infty}(1 - \eta\Delta t)^n D(x_t) = 0$, i.e., x_t exponentially converges to $x_d(T)$. ∎

Proof of Proposition 6: From Proposition 3, we have $\dot{D} \leq -\eta D$ for $\dot{x} = \mathrm{GVF}(x)$. Let us consider another system $\dot{x} = f(x)$, and denote $f(x) = \mathrm{GVF}(x) + \hat{\varepsilon}(x)$, where $\|\hat{\varepsilon}(x)\| \leq \varepsilon$. Then, the following holds: $\dot{D} \leq -\eta D + \varepsilon$, since $\|\nabla D\| = 1$ and $\langle \nabla D, \varepsilon(x)\rangle \leq \varepsilon$.

Compare $D(x(t))$ with the scalar differential equation $\dot{y} = -\eta y + \varepsilon, y(0) = D(x(0))$, whose solution is $y(t) = e^{-\eta t} D(x(0)) + \frac{\varepsilon}{\eta}(1 - e^{-\eta t})$.

Since $D(x(0)) = y(0)$ and $\dot{D}(x(t)) \leq \dot{y}(t)$, it holds that $D(x(t)) \leq y(t)$ for all $t > 0$. Now, choose

$$t_{\mathrm{c}} \;=\; \begin{cases} \dfrac{1}{\eta}\ln\left(\dfrac{\eta D(x(0))}{\varepsilon}\right), & \text{if } D(x(0)) > \frac{\varepsilon}{\eta}, \\ 0, & \text{otherwise.} \end{cases}$$

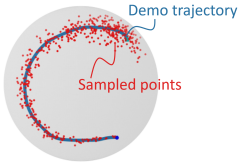

Fig. 16. Points distance Gaussian sampled on the 2-sphere. The blue line is the demonstration trajectory and the red dots are the sampled points.

Then $y(t) \leq \frac{\varepsilon}{\eta}$ for every $t \geq t_c$, and hence $D(x(t)) \leq \frac{\varepsilon}{\eta}$ for all $t > t_c$. ■

APPENDIX B
EVALUATION METRICS

Before presenting the metrics, we define a sampling scheme—*distance Gaussian sampling*—used by some of them. It draws points near the demonstration trajectory, with spread at a configuration proportional to the remaining arc length to the terminal point. In $\mathbb{R}^3$, when the demonstration is a straight line, the sampled region is a cone whose axis follows the trajectory and whose apex is the terminal point.

To achieve this, we sample configurations as follows: (i) evenly sample points $x_i = x_d(\tau_i)$ from the demonstration trajectory $x_d(\tau)$; (ii) pick one configuration in $\{x_i\}$ with probability proportional to $l(x_i, x_d(T))$, i.e., the length of the demonstration trajectory from τ_i to T; (iii) sample a point from $N(0, \alpha l(x_i, x_d(T)))$ in $T_{x_i}\mathcal{M}$ and map it to $\mathcal{M}$ using the exponential map. Fig. 16 shows points sampled by distance Gaussian sampling on S^2.

We evaluate SDSs using three metrics: (i) trajectory fitting error (FE), (ii) mimicking error (ME), and (iii) contraction rate (CR). Let the demonstration trajectory be $X_d = \{x_i, \dot{x}_i\}_{i=1}^N$.

Trajectory Fitting Error (FE). FE is the mean L2-norm of the velocity error normalized by the maximum speed $\|\dot{x}_{\max}\|$:

$$\mathrm{FE} = \frac{1}{N} \sum_i \frac{\|f(x_i) - \dot{x}_i\|}{\|\dot{x}_{\max}\|}. \tag{36}$$

Here, f is the corresponding SDS for X_d.

Mimicking Error (ME). Sample M configurations x_j using distance Gaussian sampling on X_d; ME is the mean L2-norm between model outputs and velocities parallel transported from the closest configurations:

$$\mathrm{ME} = \frac{1}{M} \sum_j \frac{\|f(x_j) - \Gamma_{x^*(x_j)}^{x_j}(\dot{x}^*(x_j))\|}{\|\dot{x}_{\max}\|}, \tag{37}$$

where $x^*(x_j)$ and $\dot{x}^*(x_j)$ are the closest point on x_d to x_j and its velocity.

Contraction Rate (CR). CR averages Lyapunov exponents of distances to x_d: sample M configurations x_j via distance Gaussian sampling; generate solution trajectories $x_j(t)$; compute $d(x_j(t), x^*(x_j(t)))$ for each; take each distance signal's Lyapunov exponent and average them.

Inference Speed (IS): For umbrella insertion, IS is measured as the inference frequency (Hz), using RTX 3090 GPU.

Success Rate (SR): For umbrella insertion, SR is the fraction of trajectories with $x(t_f)$ within a threshold distance of $x_d(T)$, with $t_f = 15$s; thresholds are 0.2 on SO(3) and 0.02 m on $\mathbb{R}^3$.

APPENDIX C
EXPERIMENT DETAILS

A. Letter Writing Task

We consider S^2 to be embedded in $\mathbb{R}^3$, which inherits the Euclidean inner product: $\langle \dot{x}, \dot{x} \rangle = \dot{x}^T \dot{x}$. For SDDM, SEDS, and ManiFlow, we use the spherical coordinate system: $x = (\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta))$.

1) Dataset: We use the LASA dataset [30] mapped to S^2, rescaled to enlarge trajectories. From 30 shapes, we select 26 single-letter shapes and, for each, one of seven demonstrations. Each trajectory contains 1000 configuration–velocity pairs. For BC-DeepOVec, we generate the CVF and MVF output data via two sampling strategies (10,000 samples each): an approximately uniform grid and GMM sampling whose component means coincide with the 1000 discretized trajectory points (standard deviation 0.1).

2) Model Architectures:

BC-DeepOVec: Identical DeepOVec architectures for c_θ and m_θ; branch networks are FCNNs with 6×1024 hidden layers (ReLU), inner product dimension 64.

SEDS: 6 Gaussian components.

SDDM: Nominal vector field: FCNN (3×512, ReLU); Lyapunov function: ICNN (3×512, smooth ReLU).

LAGS-DS: K-component GMM-weighted global LPV-DS with GPR-modulated local linear fields, where K is automatically determined by the MATLAB implementation available on the official LAGS-DS GitHub page.

ManiFlow: Bijective and main maps: 2×512 hidden layers (tanh); dynamics: 2×100 (leakyReLU).

RSDS: Scale function: FCNN (2×128, leakyReLU); vector field: FCNN (2×256, leakyReLU); 4th-order Runge–Kutta (Munthe-Kaas) with 3 steps, total length 1.

NCDS: Jacobian network: FCNN (2×512, softplus); reference $x_0 = [0,0]$, $\dot{x}_0 = [0,0]$; $\Delta t = 0.05$; $\varepsilon = 10^{-3}$.

B. Umbrella-Inserting Task in SE(3)

We view SE(3) as the product manifold $\mathrm{SO}(3) \times \mathbb{R}^3$. Let $x = (R, p)$ with $R \in \mathrm{SO}(3)$ and $p \in \mathbb{R}^3$, and let $\dot{x} = (\dot{R}, \dot{p}) \in T_x\mathrm{SE}(3)$ with $\dot{R} \in T_R\mathrm{SO}(3)$. We equip $T_x\mathrm{SE}(3)$ with the weighted inner product $\langle \dot{x}, \dot{x} \rangle := \mathrm{tr}(\dot{R}^\top \dot{R}) + 100\, \dot{p}^\top \dot{p}$. For SDDM, SEDS, and ManiFlow, SO(3) is parameterized locally by its Lie algebra.

We use an Nvidia RTX 3090 throughout the experiments.

1) Datasets: We collect five human-demonstrated joint trajectories of umbrella insertion, each targeting a different hole of the umbrella holder. From these, we derive SE(3) trajectories, each with 501 configuration–velocity pairs. For BC-DeepOVec, we generate the CVF and MVF output data via two sampling strategies: (1) uniform sampling (100,000 samples); and (2) GMM sampling (900,000 samples) whose component means coincide with the 501 discretized-trajectory points (standard deviation 0.05 in $\mathbb{R}^3$ and 0.1 in SO(3)).

2) Model Architectures:

BC-DeepOVec: Branch networks are FCNNs (6 layers × 1024 units, ReLU), inner-product dimension 64.

SEDS: Five models with Gaussian components $k \in \{4,5,6,8,9\}$.

SDDM: Nominal field: FCNN (4×1024, ReLU); Lyapunov: ICNN (4×1024, smooth ReLU) [8].

LAGS-DS: K-component GMM-weighted global LPV-DS with GPR-modulated local linear fields, where K is automatically determined by the MATLAB implementation available on the official LAGS-DS GitHub page.

ManiFlow: Bijective/main maps: 2×512 (tanh); dynamics: 2×100 (leakyReLU).

RSDS: Scale: FCNN (3×512, leakyReLU); diffeomorphism field: FCNN (4×1024, leakyReLU); integration: 4th-order Runge–Kutta (Munthe–Kaas), 5 steps, total time $= 1$.

NCDS: We use the latent version, as in the original paper (Algorithm 1) [46]. VAE: Injective VAE (x_dim=6, z_dim=3, $\beta = 1$) with 3 rational quadratic spline flows (bins=8, tail bound=5.0); Jacobian network: FCNN (2×512, softplus) $x_0 = [0,0,0]$, $\dot{x}_0 = [0,0,0]$; $\Delta t = 0.05$; $\varepsilon = 10^{-3}$.

3) Robot Control: Given an initial pose $x(0)$ and a learned vector field $f : \mathcal{M} \to T\mathcal{M}$ with $f(x) \in T_x\mathcal{M}$, we command the end-effector body twist via the left-trivialization $\mathcal{V} = x^{-1} f(x)$. Joint velocities are computed as

$$\dot{\theta} = J^{\dagger}(\theta)\,\mathcal{V}, \tag{38}$$

where $J^{\dagger}(\theta)$ is the pseudoinverse of the geometric Jacobian. For redundant robots (e.g., the 7-DoF Franka Emika Panda), (38) yields the minimum-ℓ_2-norm solution $\arg\min_{\dot{\theta}} \{\|\dot{\theta}\|_2 : J(\theta)\dot{\theta} = \mathcal{V}\}$. Control commands are updated immediately after model inference; thus the control rate is bounded by the model's inference rate.

References

[1] P. Y. Li and R. Horowitz, "Passive velocity field control of mechanical manipulators," *IEEE Trans. Robot. Autom.*, vol. 15, no. 4, pp. 751–763, 1999.

[2] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, 2011.

[3] A. Lemme, K. Neumann, R. F. Reinhart, and J. J. Steil, "Neural learning of vector fields for encoding stable dynamical systems," *Neurocomput.*, vol. 141, pp. 3–14, 2014.

[4] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robot. Auton. Syst.*, vol. 70, pp. 1–15, 2015.

[5] C. Blocher, M. Saveriano, and D. Lee, "Learning stable dynamical systems using contraction theory," in *Int. Conf. Ubiquitous Robots Ambient Intell. (URAI)*, pp. 124–129, 2017.

[6] M. Saveriano and D. Lee, "Incremental skill learning of stable dynamical systems," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 6574–6581, 2018.

[7] C. Briat, "Linear parameter-varying and time-delay systems," *Anal. Observ. Filter. Control*, vol. 3, no. 5-7, p. 254, 2014.

[8] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *Int. Conf. Mach. Learn.*, pp. 146–155, 2017.

[9] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 3964–3979, 2020.

[10] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[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 *Learn. Dyn. Control*, pp. 630–639, 2020.

[12] S. Calinon, "Gaussians on riemannian manifolds: Applications for robot learning and adaptive control," *IEEE Robot. Autom. Mag.*, vol. 27, no. 2, pp. 33–45, 2020.

[13] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 5231–5237, 2020.

[14] J. Urain, D. Tateo, and J. Peters, "Learning stable vector fields on lie groups," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 12569–12576, 2022.

[15] J. Zhang, H. B. Mohammadi, and L. Rozo, "Learning riemannian stable dynamical systems via diffeomorphisms," in *Conf. Robot Learn.*, pp. 1211–1221, 2023.

[16] L. Lu, P. Jin, and G. E. Karniadakis, "Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," *arXiv:1910.03193*, 2019.

[17] C. Yang, C. Chen, N. Wang, Z. Ju, J. Fu, and M. Wang, "Biologically inspired motion modeling and neural control for robot learning from demonstrations," *IEEE Trans. Cogn. Dev. Syst.*, vol. 11, no. 2, pp. 281–291, 2018.

[18] D. A. Duque, F. A. Prieto, and J. G. Hoyos, "Trajectory generation for robotic assembly operations using learning by demonstration," *Robot. Comput.-Integr. Manuf.*, vol. 57, pp. 292–302, 2019.

[19] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," in *Int. Joint Conf. Auton. Agents Multiagent Syst.*, pp. 1–8, 2007.

[20] M. Schneider and W. Ertel, "Robot learning by demonstration with local gaussian process regression," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 255–260, 2010.

[21] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," in *Conf. Robot Learn.*, pp. 37–46, 2017.

[22] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Adv. Neural Inf. Process. Syst.*, vol. 26, 2013.

[23] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *Int. J. Robot. Res.*, vol. 38, no. 7, pp. 833–852, 2019.

[24] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter-and extrapolation capabilities," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 4301–4308, 2019.

[25] N. Jaquier, D. Ginsbourger, and S. Calinon, "Learning from demonstration with model-based gaussian process," in *Conf. Robot Learn.*, pp. 247–257, 2020.

[26] B. Lee, Y. Lee, S. Kim, M. Son, and F. C. Park, "Equivariant motion manifold primitives," in *Conf. Robot Learn.*, 2023.

[27] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy, "Task-conditioned variational autoencoders for learning movement primitives," in *Conf. Robot Learn.*, pp. 933–944, 2020.

[28] Y. Lee, B. Lee, S. Kim, and F. C. Park, "Motion manifold flow primitives for task-conditioned trajectory generation under complex task-motion dependencies," *arXiv:2407.19681*, 2024.

[29] Y. Lee, "Mmp++: Motion manifold primitives with parametric curve models," *IEEE Trans. Robot.*, 2024.

[30] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robot. Auton. Syst.*, vol. 62, no. 6, pp. 752–765, 2014.

[31] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *arXiv:2102.03861*, 2021.

[32] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural Comput.*, vol. 25, no. 2, pp. 328–373, 2013.

[33] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 2, pp. 752–757, 2001.

[34] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 958–963, 2002.

[35] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control—a unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.

[36] A. Pervez, A. Ali, J.-H. Ryu, and D. Lee, "Novel learning from demonstration approach for repetitive teleoperation tasks," in *IEEE World Haptics Conf. (WHC)*, pp. 60–65, 2017.

[37] Y. Fanger, J. Umlauft, and S. Hirche, "Gaussian processes for dynamic movement primitives with application in knowledge-based cooperation," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 3913–3919, 2016.

[38] J. Umlauft, Y. Fanger, and S. Hirche, "Bayesian uncertainty modeling for programming by demonstration," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 6428–6434, 2017.

[39] A. Pervez, Y. Mao, and D. Lee, "Learning deep movement primitives using convolutional neural networks," in *IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, pp. 191–197, 2017.

[40] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven lyapunov candidates," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 1216–1222, 2013.

[41] N. B. Figueroa Fernandez and A. Billard, "A physically-consistent bayesian non-parametric mixture model for dynamical system learning," *Proc. Mach. Learn. Res.*, 2018.

[42] V. Sindhwani, S. Tu, and M. Khansari, "Learning contracting vector fields for stable imitation learning," *arXiv:1804.04878*, 2018.

[43] F. Nawaz, T. Li, N. Matni, and N. Figueroa, "Learning complex motion plans using neural odes with safety and stability guarantees," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 17216–17222, 2024.

[44] A. Abyaneh, M. G. Boroujeni, H.-C. Lin, and G. Ferrari-Trecate, "Contractive dynamical imitation policies for efficient out-of-sample recovery," *arXiv:2412.07544*, 2024.

[45] S. Jaffe, A. Davydov, D. Lapsekili, A. K. Singh, and F. Bullo, "Learning neural contracting dynamics: Extended linearization and global guarantees," *Adv. Neural Inf. Process. Syst.*, vol. 37, pp. 66204–66225, 2024.

[46] H. B. Mohammadi, S. Hauberg, G. Arvanitidis, N. Figueroa, G. Neumann, and L. Rozo, "Neural contractive dynamical systems," in *Int. Conf. Learn. Represent. (ICLR)*, 2024.

[47] N. Figueroa and A. Billard, "Locally active globally stable dynamical systems: Theory, learning, and experiments," *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 312–347, 2022.

[48] J. R. Medina and A. Billard, "Learning stable task sequences from demonstration with linear parameter varying systems and hidden markov models," in *Conf. Robot Learn.*, pp. 175–184, 2017.

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

[50] H. C. Ravichandar, I. Salehi, and A. P. Dani, "Learning partially contracting dynamical systems from demonstrations.," in *CoRL*, pp. 369–378, 2017.

[51] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annu. Rev. Control*, vol. 52, pp. 135–169, 2021.

[52] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1749–1767, 2023.

[53] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," in *Conf. Robot Learn.*, pp. 1519–1539, 2021.

[54] H. Tsukamoto and S.-J. Chung, "Neural contraction metrics for robust estimation and control: A convex optimization approach," *IEEE Control Syst. Lett.*, vol. 5, no. 1, pp. 211–216, 2020.

[55] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Syst. Control Lett.*, vol. 96, pp. 51–59, 2016.

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

[57] R. Pérez-Dattari, C. Della Santina, and J. Kober, "Puma: Deep metric imitation learning for stable motion primitives," *Adv. Intell. Syst.*, vol. 6, no. 11, p. 2400144, 2024.

[58] S. Sun and N. Figueroa, "Se (3) linear parameter varying dynamical systems for globally asymptotically stable end-effector control," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 5152–5159, 2024.

[59] L. Tang and R. G. Landers, "Multiaxis contour control—the state of the art," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 1997–2010, 2013.

[60] W. Yao, H. G. de Marina, B. Lin, and M. Cao, "Singularity-free guiding vector field for robot navigation," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1206–1221, 2021.

[61] Y. A. Kapitanyuk, A. V. Proskurnikov, and M. Cao, "A guiding vector-field algorithm for path-following control of nonholonomic mobile robots," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 4, pp. 1372–1385, 2017.

[62] W. Yao, B. Lin, B. D. Anderson, and M. Cao, "Topological analysis of vector-field guided path following on manifolds," *IEEE Trans. Autom. Control*, vol. 68, no. 3, pp. 1353–1368, 2022.

[63] M. P. Do Carmo and J. Flaherty Francis, *Riemannian geometry*, vol. 6. 1992.

[64] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.

[65] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," 1995.

[66] W. Wang and J.-J. E. Slotine, "On partial contraction analysis for coupled nonlinear oscillators," *Biol. Cybern.*, vol. 92, no. 1, pp. 38–53, 2005.

[67] K. M. Lynch and F. C. Park, *Modern robotics*. 2017.

BIOGRAPHY

Byeongho Lee received the B.S. degree from Yonsei University, Seoul, South Korea, in 2019, and the M.S. and Ph.D. degrees from Seoul National University, Seoul, in 2021 and 2025, respectively, all in mechanical engineering.

He is currently a Researcher with the Future Robotics Office, Samsung Electronics, Seoul, South Korea. His research interests include robot learning, learning from demonstration, and geometric machine learning.

Yonghyeon Lee (Member, IEEE) received the B.S. degree in mechanical engineering (major) and physics (minor) and the Ph.D. degree in mechanical engineering (with a focus on robotics and AI) from Seoul National University, Seoul, South Korea, in 2018 and 2023, respectively.

From 2023 to 2025, he was an AI Research Fellow with the Center for Artificial Intelligence and Natural Sciences, Korea Institute for Advanced Study, Seoul. He is currently a Postdoctoral Associate with the Biomimetic Robotics Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, where he has been working since March 2025. His research interests include geometric data analysis and robot planning, learning, and control.

Junsu Ha (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering in 2022 and 2024, respectively, from Seoul National University, Seoul, South Korea, where he is currently working toward the Ph.D. degree.

His research interests include robot learning, deep learning, motion planning, and learning from demonstration.

Frank C. Park (Fellow, IEEE) received the B.S. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1985, and the Ph.D. degree in applied mathematics from Harvard University, Cambridge, MA, USA, in 1991.

He is currently a Professor of mechanical engineering with Seoul National University (SNU), Seoul, South Korea, with joint appointments in the SNU Graduate School of AI and the SNU Robotics Institute. Before joining SNU, he served on the Faculty of Mechanical and Aerospace Engineering, University of California, Irvine, CA, USA. He has held visiting positions at the HKUST Robotics Institute, Hong Kong, Georgia Tech Interactive Computing Department, Atlanta, GA, USA, and the NYU Courant Institute, New York, NY, USA. His research interests include robot motion and manipulation planning, mathematical data science and machine learning, and related areas of applied mathematics.

Dr. Park is former Editor-In-Chief of the IEEE TRANSACTIONS ON ROBOTICS (2013–2018), past President of the IEEE Robotics and Automation Society (2022–2023), IEEE Division X Director-elect (2026), and Fellow of the Korean Academy of Science and Technology (KAST) and the National Academy of Engineering of Korea (NAEK). He is author of Modern Robotics (with K. Lynch, Cambridge U. Press 2017) and Dynamics of Mechanical Systems (with Kyu-Min Park, Cambridge U. Press 2025), and developer of the EdX Course Robot Mechanics and Control I–II.