PUMA

Deep Metric Imitation Learning for Stable Motion Primitives

Pérez-Dattari, Rodrigo; Della Santina, Cosimo; Kober, Jens

DOI
10.1002/aisy.202400144

Publication date
2024

Document Version
Final published version

Published in
Advanced Intelligent Systems

Citation (APA)
Pérez-Dattari, R., Della Santina, C., & Kober, J. (2024). PUMA: Deep Metric Imitation Learning for Stable Motion Primitives. *Advanced Intelligent Systems*, *6*(11), Article 2400144. https://doi.org/10.1002/aisy.202400144

PUMA: Deep Metric Imitation Learning for Stable Motion Primitives

Rodrigo Pérez-Dattari, Cosimo Della Santina, and Jens Kober*

Imitation learning (IL) facilitates intuitive robotic programming. However, ensuring the reliability of learned behaviors remains a challenge. In the context of reaching motions, a robot should consistently reach its goal, regardless of its initial conditions. To meet this requirement, IL methods often employ specialized function approximators that guarantee this property by construction. Although effective, these approaches come with some limitations: 1) they are typically restricted in the range of motions they can model, resulting in suboptimal IL capabilities, and 2) they require explicit extensions to account for the geometry of motions that consider orientations. To address these challenges, we introduce a novel stability loss function that does not constrain the function approximator's architecture and enables learning policies that yield accurate results. Furthermore, it is not restricted to a specific state space geometry; therefore, it can easily incorporate the geometry of the robot's state space. Proof of the stability properties induced by this loss is provided and the method is empirically validated in various settings. These settings include Euclidean and non-Euclidean state spaces, as well as first-order and second-order motions, both in simulation and with real robots. More details about the experimental results can be found at https://youtu.be/ZWKLGntCI6w.

1. Introduction

Imitation learning (IL) provides a powerful framework for the intuitive programming of robotic systems. Its strength lies in its ability to leverage human-like learning methodologies, such as demonstrations and corrections, making it accessible to non-robotics experts. This attribute significantly reduces the resources needed to build robotic systems. However, the data-driven nature of these methodologies presents a challenge: providing guarantees about the learned behaviors.

In the context of reaching motions, it is crucial for the robot's motions to consistently reach the intended target, irrespective of the robot's initial conditions. Thus, modeling motions as dynamical systems proves beneficial. This approach turns the problem into a question of ensuring *global asymptotic stability* (or stability, for short) at the goal, and tools from dynamical system theory can then be applied to guarantee this property.

Numerous methods have been proposed to ensure stability in motions represented by dynamical systems. However, they often exhibit at least one of the following limitations: 1) constraining the structure of their function approximators and/or 2) being designed with the assumption that the robot's state space is Euclidean. This can be elaborated as follows.

Constrained function approximators. To ensure stability guarantees, methods often constrain the structure of their function approximators. For instance, some approaches necessitate invertibility,[1–3] while others require positive or negative definiteness.[4–6] However, these methods do not enable the full exploitation of modern deep neural network (DNN) architectures, as these constraints are not typically present in DNNs. This limitation hinders their broader application in more complex models, where integrating these constraints is challenging. Furthermore, inherently constraining function approximators can overly restrict the range of solutions to which they can converge, resulting in less flexible models than necessary. This leads to suboptimal IL capabilities. In our context, this problem is known as the stability versus accuracy dilemma.[7]

Euclidean assumption. Learned motions must integrate with the geometry of the space used to represent a robot's state. For example, the end-effector of a manipulator should always reach the intended target in both position and orientation space. However, orientations are often represented in non-Euclidean spaces, such as $SO(3)$ or $\mathcal{S}^3$. This is because Euclidean representations like Euler angles may not always provide a continuous description of motions that are inherently continuous[8] (This issue stems from singularities and nonunique representations). Such continuity is often a requirement for motions modeled as dynamical systems. Furthermore, the generalization capabilities of function approximations are compromised by noncontinuous representations. To enable proper generalization, states that are close in the real world should be represented as closely related in the function approximator's input, that is, the state space representation.

R. Pérez-Dattari, C. Della Santina, J. Kober
Department of Cognitive Robotics
Delft University of Technology
Mekelweg 2, 2628 CD Delft, The Netherlands
E-mail: r.j.perezdattari@tudelft.nl

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202400144.

DOI: 10.1002/aisy.202400144

Nevertheless, previous methods for stable motion generation have been initially designed under the assumption that the state space is Euclidean.[2–4,9] Consequently, some have later been explicitly adapted to account for the geometry of motions that consider orientations,[10–12] resulting in rather convoluted learning frameworks.

In this work, we present a DNN framework capable of learning accurate, stable motions in state spaces with arbitrary geometries without constraining the DNN's architecture. To accomplish this, we introduce a novel loss function that repurposes the triplet loss, commonly used in deep metric learning literature.[13,14] We prove that this loss imposes conditions on the DNN's latent space that enforce the learned dynamical system to have a globally asymptotically stable equilibrium at the motion's goal state (see **Figure** 1). To account for the geometry of the state space, it is sufficient to consider its corresponding metric during the computation of the loss, the choice of which depends on the specific task at hand. We validate our method in various settings, including Euclidean, non-Euclidean, first-order, and second-order motions. Additionally, real-world experiments controlling the six-dimensional pose (x–y–z position and unit quaternion orientation) of a robot manipulator's end effector demonstrate the method's practical applicability and potential.

The rest of this paper offers a thorough discussion of our developments. Following a review of the relevant literature, we delve into foundational concepts and problem formulation. We then present the details of our methodology, provide proof regarding the stability of the learned motion, and discuss the integration of the triplet loss function within the context of non-Euclidean state representations. We validate our approach through several experiments and conclude by considering potential directions for future research based on our findings.

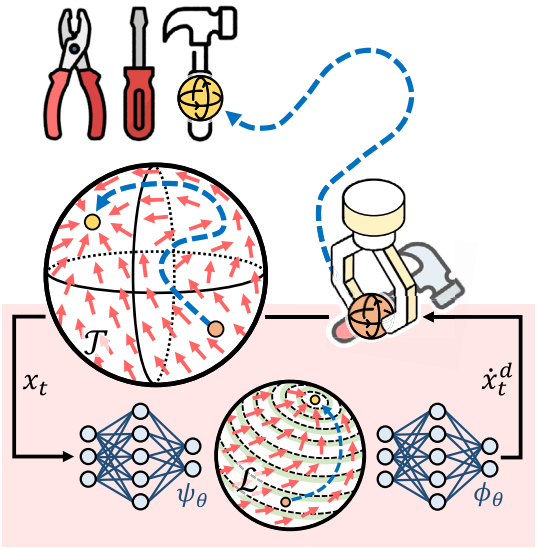


Figure 1. Motion learned using the proposed framework. The blue trajectory in the task space $\mathcal{T}$ demonstrates the evolution of the robot's end effector state x_t when represented in a spherical manifold. The evolution of this trajectory is governed by the dynamical system $\dot{x}_t = \phi_\theta(\psi_\theta(x_t))$, depicted as a vector field of red arrows in the remaining of the space. Through deep metric learning, this system is stabilized by deriving a simpler representation in the latent space $\mathcal{L}$.

2. Related Works

Three categories of works are relevant to this paper. First, there are papers that focus on learning stable motions, assuming these motions occur in Euclidean state spaces. Second, others explore learning motions in non-Euclidean state spaces, but do not consider the stability of the learned motions. Finally, a third set of papers addresses both learning motions in non-Euclidean state spaces and their stability. Importantly, in every case, works use either time-varying (non-autonomous) or time-invariant (autonomous) dynamical systems for motion modeling. In time-varying systems, evolution explicitly depends on time (or a phase). Conversely, time-invariant systems do not directly depend on time; instead, they rely on their time-varying input (i.e., the state of the system). The property of a system being time invariant or not dictates the strategies we can use to ensure its stability. Thus, making a distinction between these systems is important. Notably, both types of formulations have been shown to be complementary in the context of IL.[15,16]

This work focuses on time-invariant dynamical systems. Consequently, although we consider both methodologies for motion learning, we delve deeper into the literature on time-invariant systems. Furthermore, while we concentrate on methods that ensure asymptotic stability, we acknowledge that there are studies imposing other conditions, such as via-point conditioning,[17,18] which can be significantly relevant in some robotic contexts.

2.1. Stability in Euclidean State Spaces

Regarding time-varying dynamical systems, a seminal work in IL that addresses the problem of learning stable motions introduces Dynamical Movement Primitives (DMPs).[19] DMPs take advantage of the time dependency (via the phase of the canonical system) of the dynamical system to make it evolve into a simple and well-understood system as time goes to infinity. The simple system is designed to be stable by construction. As a consequence, the stability of the learned motions can be guaranteed. This concept has been extended in multiple ways, for instance through probabilistic formulations,[20,21] or adapted to the context of DNNs.[22–24]

Conversely, IL approaches based on time-invariant dynamical systems often constrain the function approximator used to model the dynamical systems. Such constraints ensure stability by construction. One seminal work introduces the Stable Estimator of Dynamical Systems (SEDS).[4] This approach imposes constraints on the structure of a Gaussian Mixture Regression (GMR) such that the conditions for Lyapunov stability are always met. Multiple extensions of SEDS have been proposed, for instance by using physically consistent priors,[25] contraction theory,[26] or diffeomorphisms.[27]

Other works propose explicitly learning Lyapunov functions that are consistent with the demonstrations. These functions are then used to correct the transitions learned by the dynamical systems, ensuring that they are always stable according to the

learned Lyapunov functions.[5,6,28] Moreover, some papers have employed concepts such as contraction metrics[29,30] and diffeomorphisms[1–3,31] to impose stability, in the sense of Lyapunov, in time-invariant dynamical systems.

Understandably, all of these methods constrain some part of their learning framework to ensure stability. From one perspective, this is advantageous as it guarantees stability. However, in many cases, this comes at the expense of reduced accuracy in the learned motions. Notably, some recent methods have managed to mitigate this loss in accuracy.[2,3] Nevertheless, they are still limited in terms of the family of models that can be used with these frameworks, which harms their scalability.

In previous work,[9] we addressed this issue by employing tools from the deep metric learning literature.[14] There, we introduced CONDOR, which uses a contrastive loss to enforce stability in learned motions through the optimization process of a DNN. This approach proved effective in learning stable, accurate, and scalable motions from human demonstrations. However, this method presents two important limitations. First, it requires the design of a stable and well-understood system for computing the contrastive loss (referred to as $f^{\mathscr{L}}$ in Section 3.3), which can significantly impact learning performance. Second, similar to the other methods described in this subsection, CONDOR is limited to operating within Euclidean state spaces.

In this work, we introduce a novel deep metric learning loss that ensures stability while addressing the limitations of,[9] specifically, the need for the function $f^{\mathscr{L}}$ and the inability to learn motions in non-Euclidean manifolds. Moreover, this loss requires weaker conditions for imposing stability, leading to, for instance, more robust performance when learning second-order dynamical systems.

2.2. Stability in Non-Euclidean State Spaces

As noted previously, it is crucial to consider the geometry of our state spaces when learning motions requiring pose control. Consequently, many studies have adapted methods that originally assumed data from Euclidean spaces to work with data from non-Euclidean manifolds, such as $SO(3)$ and $\mathcal{S}^3$. In the context of IL, pioneering approaches utilized time-varying dynamical systems (extensions of DMPs) to incorporate the geometry of orientation representations into their models.[8,32,33] Because DMPs inherently ensure stability, these methods are stable as well.

In contrast, although several geometry-aware IL approaches based on time-invariant dynamical systems have been introduced, such as Gaussian process regressions,[34,35] GMRs,[36–39] and kernelized movement primitives,[40,41] such methods are not inherently stable. As a result, in these cases, models need to be explicitly endowed with stability properties. This limitation has been addressed in recent works, where ref. [42] uses GMRs and employs contraction theory to ensure stability considering the robot's orientation geometry. Furthermore, recent methods have extended the use of diffeomorphism-based techniques for stability to generate motions in non-Euclidean manifolds as well.[10–12]

These diffeomorphism-based methods are of particular interest to our work, as our method is grounded in similar concepts for achieving stability. In both approaches, we transfer the stability properties of a simple system in the latent space of a DNN to the dynamical system that models motion in task space. However, unlike these methods, our approach learns this property, whereas the other works constrain the DNN structure to ensure its satisfaction. Moreover, our method is not constrained to diffeomorphic solutions for transferring the stability properties of the simple system to task space. Lastly, our method allows us to seamlessly incorporate the geometrical aspects of the robot's state space into the learned model, as this integration is factored into the DNN's optimization process.

3. Preliminaries

3.1. Dynamical Systems for Reaching Tasks

In this work, we model motions as nonlinear time-invariant dynamical systems represented by

$$\dot{x}_t = f(x_t) \tag{1}$$

where $x_t \in \mathcal{T}$, with $\mathcal{T} \subseteq \mathbb{R}^n$ being the task space, is the robot's state and $f : \mathcal{T} \to \mathbb{R}^n$ is a differentiable function. Additionally, the subscript t indicates the time instance to which the state corresponds. Since we are interested in solving reaching tasks, we aim to construct systems with a globally asymptotically stable equilibrium at a goal state x_g. This implies that

$$\lim_{t\to\infty} ||x_g - x_t|| = 0, \quad \forall x_t \in \mathcal{T} \tag{2}$$

3.2. Problem Formulation

We consider a robot learning a reaching motion in the space $\mathcal{T}$ toward the goal state $x_g \in \mathcal{T}$. Based on a set of demonstrations D, the robot is expected to imitate the behavior shown in the demonstrations while always reaching x_g, regardless of its initial state. The dataset D contains N trajectories τ . These trajectories show the evolution of a dynamical system's state x_t when starting from some initial condition x_0.

We assume that the demonstrations are drawn from an optimal distribution over trajectories, denoted as $p^*(\tau)$, that adheres to the optimal dynamical system f^*. Here, "optimal" refers to the behavior that the demonstrator deems best.

The robot's motion follows the parametrized system $f_\theta^{\mathcal{T}}$, inducing the distribution $p_\theta(\tau)$, where θ is a parameter vector. Then, the objective is to find the vector θ^* that minimizes the difference, expressed as the (forward) Kullback–Leibler divergence, between $p_\theta(\tau)$ and $p^*(\tau)$, while ensuring that x_g is a globally asymptotically stable equilibrium.

$$\theta^* = \arg\min_{\theta\in\Theta} D_{\mathcal{KL}}(p^*(\tau)||p_\theta(\tau)) \tag{3a}$$

$$\text{s.t.} \quad \lim_{t\to\infty} d(x_g, x_t) = 0, \quad \forall x_t \in \mathcal{T} \tag{3b}$$

Here, Θ is the parameter space of a DNN, and $d(\cdot,\cdot)$ is a distance function.

3.3. Stability Conditions

In,[9] the stability conditions are formulated to enforce stability in $f_\theta^{\mathcal{T}}$. This is achieved by ensuring that $f_\theta^{\mathcal{T}}$ inherits the stability properties of a simple and stable system, referred to as $f^{\mathcal{L}}$. Thus, if $f^{\mathcal{L}}$ is asymptotically stable, $f_\theta^{\mathcal{T}}$ will also be asymptotically stable. To accomplish this, the system $f^{\mathcal{L}}$ is designed to define the evolution of a state y_t with an initial condition derived from mapping x_0 to the output of a hidden layer in the DNN that parameterizes $f_\theta^{\mathcal{T}}$. As a result, the dynamical system $f_\theta^{\mathcal{T}}$ is expressed as a composition of two functions, ψ_θ and ϕ_θ

$$\dot{x}_t = f_\theta^{\mathcal{T}}(x_t) = \phi_\theta(\psi_\theta(x_t)) \tag{4}$$

Here, $f_\theta^{\mathcal{T}}$ is a standard DNN with $\mathcal{L}$ layers. ψ_θ denotes layers $1, \ldots, l$, and ϕ_θ layers $l+1, \ldots, L$. We define the output of layer l as the latent space $\mathcal{Z}_{\mathbb{R}^m}$ (Figure 1), which is where y_t resides, that is, $y_t \in \mathcal{Z}$. Note that $\mathcal{Z}_{\mathbb{R}^m}$ implies the dimensionality of the vectors used to represent $\mathcal{T}$ and $\mathcal{Z}$ does not need to be the same. Moreover, for simplicity, although we use the same θ notation for both ψ_θ and ϕ_θ, each symbol actually refers to a different subset of parameters within θ. These subsets together form the full parameter set in $f_\theta^{\mathcal{T}}$.

Additionally, we introduce a third dynamical system. This system denotes the evolution in $\mathcal{Z}$ of the states visited by $f_\theta^{\mathcal{T}}$ when mapped using ψ_θ, which yields the relationship

$$\dot{y}_t = f_\theta^{\mathcal{T}\to\mathcal{Z}}(x_t) = \frac{\partial \psi_\theta(x_t)}{\partial t} \tag{5}$$

Figure 2 provides an example of the introduced dynamical systems.

Then, the stability conditions of[9] can be written as follows.

Theorem 1 (Stability conditions: v1). *Let $f_\theta^{\mathcal{T}}, f_\theta^{\mathcal{T}\to\mathcal{Z}}$ and $f^{\mathcal{Z}}$ be the introduced dynamical systems. Then, in the region $\mathcal{T}$, x_g is a globally asymptotically stable equilibrium of $f_\theta^{\mathcal{T}}$ if, $\forall x_t \in \mathcal{T}$:*

1) $f_\theta^{\mathcal{T}\to\mathcal{Z}}(x_t) = f^{\mathcal{Z}}(y_t)$
2) $\psi_\theta(x_t) = y_g \Rightarrow x_t = x_g$

These conditions imply that if the system $f_\theta^{\mathcal{T}\to\mathcal{Z}}$ behaves like $f^{\mathcal{Z}}$, and any point other than x_g is excluded from mapping to the latent goal y_g, then $f_\theta^{\mathcal{T}}$ is stable. Note that the latent goal is defined by the mapping $\psi_\theta(x_g) = y_g$.

In this work, we reformulate these conditions and propose a novel way to optimize them. This adaptation endows the learning framework with increased flexibility, enabling it to tackle a wider array of problems (e.g., non-Euclidean state spaces) and achieve improved performance.

3.4. Deep Metric Learning: The Triplet Loss

Commonly employed in the Deep Metric Learning literature, the triplet loss[13] has been utilized for learning and structuring latent state representations.[14] Its function is to cluster similar observations together and differentiate dissimilar ones within the latent space of a DNN. In this work, however, the triplet loss is used differently. Although we continue to use this loss to impose a certain structure on the DNN's latent space, its purpose

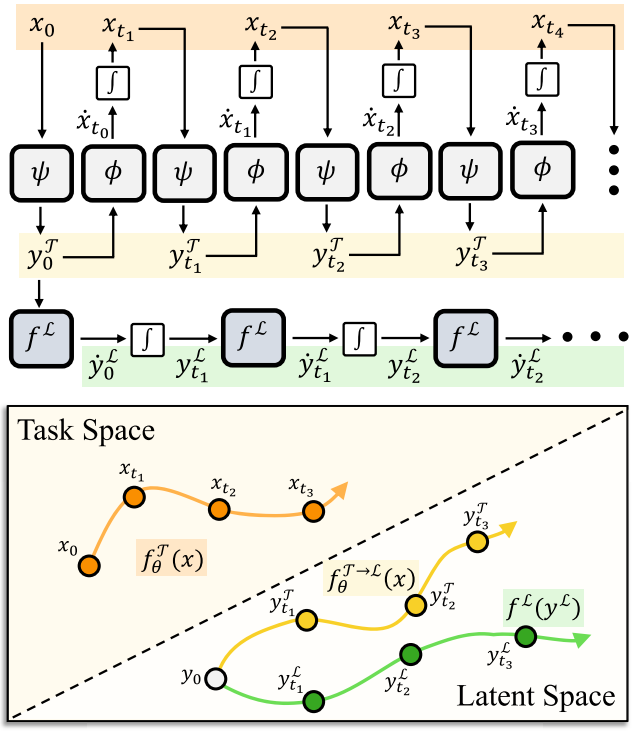


Figure 2. Example of trajectories generated by simulating the systems $f_\theta^{\mathcal{T}}$, $f_\theta^{\mathcal{T}\to\mathcal{Z}}$, and $f^{\mathcal{Z}}$ for different time instants. The stability conditions are not met in this case, as $f_\theta^{\mathcal{T}\to\mathcal{Z}}$ differs from $f^{\mathcal{Z}}$.

is to enforce the stability conditions, thereby ensuring that $f_\theta^{\mathcal{T}}$ is stable.

Let us recall the triplet loss

$$\ell_{\text{triplet}} = \max(0, m + d(a,p) - d(a,n)) \tag{6}$$

where $m \in \mathbb{R}_{>0}$ is the margin, a is the anchor sample, p the positive sample, and n the negative sample This loss enforces positive samples to be at least m distance closer to the anchor than the negative samples. Notably, this is enough for the loss to become zero, that is, it does not require the anchor and positive samples to have the same value.

3.5. Stability Analysis Through Comparison Functions

In this work, we employ comparison functions, namely class- $\mathcal{KL}$ functions, to prove *global asymptotic stability* at the equilibrium x_g. These functions are used to formulate a general approach for stability analysis in the sense of Lyapunov. As described in[43,44] these functions are defined as follows.

Definition 1 (class-$\mathcal{K}$ function). A continuous function $\alpha : [0, a) \to \mathbb{R}_{\geq 0}$, for $a \in \mathbb{R}_{>0}$, is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$.

Definition 2 [class-$\mathcal{L}$ function] (Note that the symbol $\mathcal{L}$ is used in two distinct contexts. While it represents the class-$\mathcal{L}$ functions, it is also used to denote the set $\mathcal{L}$ introduced in Section 3.3. In subsequent sections of the paper, the $\mathcal{L}$ referring to class-$\mathcal{L}$ functions is exclusively used in the form $\mathcal{KL}$).

A continuous function $\sigma:\mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$, is said to belong to class $\mathscr{L}$ if it is *weakly decreasing* (We use this term to denote functions that either remain constant or strictly decrease within any interval of their domain.) and $\lim_{s\to\infty}\sigma(s) = 0$.

Definition 3 (class-$\mathcal{KL}$ function). A function $\beta:[0,a)\times\mathbb{R}_{\geq 0}\to\mathbb{R}_{\geq 0}$, for $a\in\mathbb{R}_{>0}$, is said to belong to class-$\mathcal{KL}$ if: 1) for each fixed s, the mapping $\beta(r,s)$ belongs to class-$\mathcal{K}$ with respect to r, 2) for each fixed r, the mapping $\beta(r,s)$ belongs to class-$\mathscr{L}$ with respect to s.

Then, we can describe *global asymptotic stability* in terms of class-$\mathcal{KL}$ functions.

Theorem 2 (*Global asymptotic stability* with class-$\mathcal{KL}$ functions) *The state x_g is a globally asymptotically stable equilibrium of* (1) *in $\mathcal{T}$ if there exists a class-$\mathcal{KL}$ function β such that, $\forall t\in\mathbb{R}_{\geq 0}$ and $\forall x_0\in\mathcal{T}$*

$$||x_g - x_t|| \leq \beta(||x_g - x_0||, t) \tag{7}$$

Note that this theorem seamlessly integrates the concepts of *stability* and *attractivity* within a single function β, which acts as an upper bound of $||x_g - x_t||$. As the initial condition of the system moves further away from x_g, β correspondingly increases. Moreover, as the system evolves over time and β decreases, it follows that the system's distance to x_g will eventually decrease.

4. Methodology

We introduce the Policy via neUral Metric leArning (PUMA) framework, which learns motion primitives from human demonstrations parametrized as the dynamical system $f_\theta^{\mathcal{T}}$. Furthermore, it enforces the goal of the motion x_g to be a globally asymptotically stable equilibrium of $f_\theta^{\mathcal{T}}$ while maintaining accuracy with respect to the demonstrations. To achieve this, we augment the IL problem with the stability-enforcing loss ℓ_{stable}. Hence, our framework minimizes the loss

$$\ell_{\text{PUMA}} = \ell_{\text{IL}} + \lambda\ell_{\text{stable}} \tag{8}$$

where ℓ_{IL} is an IL loss and $\lambda\in\mathbb{R}_{>0}$ is a weight factor.

4.1. Behavioral Cloning

To minimize ℓ_{IL}, and thereby address (3Boots, B,Boots, B,Boots, B,Boots, B,Boots, B,Boots, B,a), we adopt the behavioral cloning loss used in.[9] This loss tackles (2a) by modeling the output of the deterministic dynamical system $f_\theta^{\mathcal{T}}$ as the mean of a Gaussian distribution with fixed covariance. Furthermore, it mitigates covariate shift by minimizing the multistep error over trajectory segments using backpropagation through time (BPTT), as depicted in **Figure 3**.

In every training iteration, we sample a batch $\mathscr{B}^i$ of trajectory segments $\mathscr{H}^i$ from the dataset D. These segments can start at any point within a given demonstration, with the start time defined as $t=0$. Then, by introducing the evolution function $\Phi_\theta^x(t,x_0):\mathbb{R}_{\geq 0}\times\mathcal{T}\to\mathcal{T}$, which defines the value of x_t by integrating $f_\theta^{\mathcal{T}}$ between 0 and t, with initial condition x_0, we can construct the loss

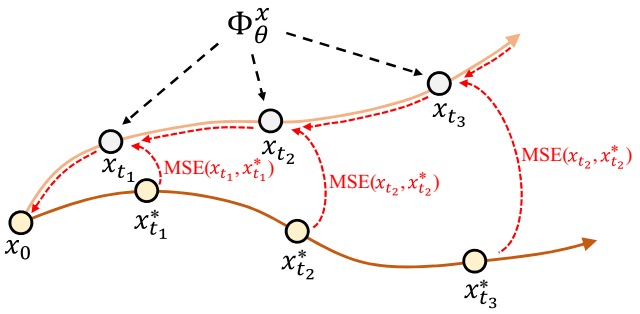


Figure 3. Illustration of the behavioral cloning loss computation. Starting from an initial condition x_0, the system $f_\theta^{\mathcal{T}}$ evolves to various time instants via Φ_θ^x. At each instant, the estimated state is compared with a demonstrated state. The red arrows show the gradient path used to update the DNN's weights using BPTT.

$$\ell_{\text{IL}} = \sum_{\mathscr{H}^i\in\mathscr{B}^i}\sum_{(t,x_t^*)\in\mathscr{H}^i} ||x_t^* - \Phi_\theta^x(t,x_0)||_2^2 \tag{9}$$

Here, the states x_t^* along the trajectory segment $\mathscr{H}^i$ serve as labels for the states predicted by the DNN from the initial condition x_0 using $\Phi_\theta^x(t,x_0)$. Note that the initial condition is obtained from $\mathscr{H}^i$, that is, $x_0 = x_0^*$.

Since we do not have an analytical solution of Φ_θ^x, we approximate it using the forward Euler method, that is

$$\Phi_\theta^x(t,x_0) = \Phi_\theta^x(t',x_0) + f_\theta^{\mathcal{T}}(\Phi_\theta^x(t',x_0))\Delta t \tag{10}$$

where $t' = t - \Delta t$ and $\Delta t\in\mathbb{R}_{>0}$ is the time step size. This integration starts with the initial state x_0, that is, $\Phi_\theta^x(0,x_0) = x_0$. It is important to note that the recursive nature of $\Phi_\theta^x(t,x_0)$ necessitates the use of BPTT for the optimization of the DNN.

4.2. Triplet Stability Loss

4.2.1. Reformulating the Stability Conditions

The stability conditions of Theorem 1 involve three dynamical systems: $f_\theta^{\mathcal{T}}$, $f_\theta^{\mathcal{T}\to\mathscr{L}}$, and $f^{\mathscr{L}}$. Note, however, that its first condition, namely, $f_\theta^{\mathcal{T}\to\mathscr{L}} = f^{\mathscr{L}}\ \forall x_t\in\mathcal{T}$, essentially states that $f_\theta^{\mathcal{T}\to\mathscr{L}}$ must exhibit *global asymptotic stability*. In other words, if the behavior of $f_\theta^{\mathcal{T}\to\mathscr{L}}$ is identical to that of another system, $f^{\mathscr{L}}$, for which stability is verified, then the stability of $f_\theta^{\mathcal{T}\to\mathscr{L}}$ is also verified. Nonetheless, if we can enforce its stability through a different method, these conditions can be more generally written as follows.

Theorem 3 (Stability conditions: v2) *Let $f_\theta^{\mathcal{T}}$ and $f_\theta^{\mathcal{T}\to\mathscr{L}}$ be the introduced dynamical systems. Then, in the region $\mathcal{T}$, x_g is a globally asymptotically stable equilibrium of $f_\theta^{\mathcal{T}}$ if, $\forall x_t\in\mathcal{T}$,*:

1) *y_g is a globally asymptotically stable equilibrium of $f_\theta^{\mathcal{T}\to\mathscr{L}}(x_t)$,*
2) *$\psi_\theta(x_t) = y_g \Rightarrow x_t = x_g$.*

4.2.2. Surrogate Stability Conditions

Theorem 3 introduces stability conditions for $f_\theta^{\mathcal{T}}$; nevertheless, it does not specify how to enforce these conditions in the system. Therefore, we introduce the surrogate stability conditions of

Theorem 3. These conditions, when met, imply that the stability conditions of Theorem 3 are also satisfied. Unlike the stability conditions, the surrogate conditions can be directly transformed into a specific loss function, ℓ_{stable}, for optimizing the DNN to enforce their satisfaction.

To formulate the surrogate stability conditions, we note that Theorem 3 can be expressed in terms of relative distances. We define the distance between any given latent state y_t and the goal state y_g as $d_t = d(y_g, y_t) = \|y_g - y_t\|$. Then, according to Condition 1 of Theorem 3, which addresses *global asymptotic stability*, the value of d_t should, generally speaking, decrease over time. Moreover, the second condition specifies that the value of d_t should remain constant only for $y_g = \psi_\theta(x_g)$. Formally, we introduce the conditions as follows.

Theorem 4 (Surrogate stability conditions) *Let two dynamical systems be governed by the equations* $\dot{x}_t = f_\theta^{\mathcal{T}}(x_t)$ *and* $\dot{y}_t = f_\theta^{\mathcal{T}\to\mathcal{L}}(y_t)$, *such that* $y_t = \psi_\theta(x_t)$. *Assume both* $f_\theta^{\mathcal{T}}$ *and* $f_\theta^{\mathcal{T}\to\mathcal{L}}$ *are continuously differentiable. Then, in the region* $\mathcal{T}$, x_g *is a globally asymptotically stable equilibrium of* $f_\theta^{\mathcal{T}}$ *if,* $\forall t \in \mathbb{R}_{\geq 0}$:

1) $d_t = d_{t+\Delta t}$, for $y_0 = y_g$,
2) $d_t > d_{t+\Delta t}$, $\forall y_0$ with $x_0 \in \mathcal{T} \setminus \{x_g\}$,

where $\Delta t \in \mathbb{R}_{>0}$.

Proof. To prove this theorem, we demonstrate that if the surrogate stability conditions are satisfied, then the stability conditions from Theorem 3 must also hold. This leads to x_g being a globally asymptotically stable equilibrium of $f_\theta^{\mathcal{T}}$.

Second Stability Condition of Theorem 3: Let us begin by analyzing the fulfillment of the second stability condition from Theorem 3. This condition states that only x_g can map to y_g via ψ_θ. Then, since y_g is defined as $\psi_\theta(x_g)$, we need to establish that no other x_t maps to y_g. To achieve this, we note that both x_t and x_0 belong to the same state space $\mathcal{T}$. Therefore, showing that this statement holds $\forall x_0 \in \mathcal{T}$ implies that it also holds $\forall x_t \in \mathcal{T}$.

Now, suppose for a contradiction that there exists some x_0 such that $x_0 \neq x_g$ and $y_0 = \psi_\theta(x_0) = y_g$. Then, according to the second surrogate condition, $d_t > d_{t+\Delta t}$, which contradicts the first surrogate condition. Consequently, the second stability condition must be satisfied.

First Stability Condition Theorem 3: Let us now study the first stability condition. Our goal is to prove that y_g is a globally asymptotically stable equilibrium of $f_\theta^{\mathcal{T}\to\mathcal{L}}$ within the region $\mathcal{L}$, where the system is defined. Surrogate condition 2 hints that the system's stability could be verified through a Lyapunov candidate defined using d_t. This is because the condition enforces the distance d_t to strictly decrease within the interval defined by Δt. However, this does not necessarily imply that the Lyapunov candidate strictly decreases with time, as it is possible for it to strictly increase locally while adhering to this condition, as exemplified in **Figure 4**. As a result, we proceed to demonstrate the *global asymptotic stability* of y_g using Theorem 2. Specifically, we aim to do so by employing a class-$\mathcal{KL}$ upper bound β that fulfills (3).

To achieve this, we first define an evolution function for the distance d_t, for a given y_0 and t, as $\delta : \mathcal{L} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, with $\delta(y_0, t) = ||y_g - \Phi_\theta^y(t, y_0)||$. Here, Φ_θ^y represents the evolution function of y_t under the dynamical system $f_\theta^{\mathcal{T}\to\mathcal{L}}$. Then, we can express the upper bound β as

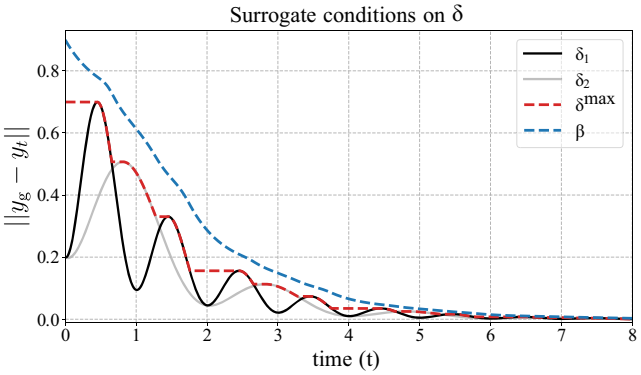


Figure 4. Time evolution of two functions δ, δ_1, and δ_2, starting with different initial conditions y_0, but same d_0. Both satisfy the surrogate stability conditions with $\Delta t = 2$. Additionally, the values of $\delta^{\max}$ and β, computed using these functions, are shown. In this representation, $\delta = e^{-a \cdot t}(\sin^2(\omega t) + d_0 \cdot \cos^2(\omega t))$ with $a = 0.75$, $d_0 = 0.2$, and $\omega = [\pi, \pi/2]$.

$$\delta(y_0, t) \leq \beta(d_0, t) \tag{11}$$

where $d_0 = \delta(y_0, 0)$. Recall that for ensuring asymptotic stability, β must also satisfy the following properties: 1) $\beta(0, t) = 0$, 2) β weakly decreases with t, 3) β is continuous with respect to d_0 and t, 4) $\beta \to 0$ as $t \to \infty$, and 5) β strictly increases with d_0.

It is important to note that verifying all these properties can lead to a lengthy proof. Thus, while we introduce β and discuss the key concepts behind its design here, the complete proof and details are provided in Prop. 1, Appendix A.

We now proceed to describe the three main aspects considered in the design of β.

Upper bound in time: First, we need to identify a β that weakly decreases with time. For this purpose, we introduce a function that computes the maximum of δ over the window $[t, t + \Delta t]$ for any given y_0 and t, that is

$$\delta_{t+\Delta t}^{\max}(y_0, t) = \max_{s \in [t,\, t+\Delta t]} \delta(y_0, s) \tag{12}$$

Clearly, this function serves as an upper bound for δ. Moreover, this function must weakly decrease with time. Considering surrogate condition 2, which indicates that $\forall s \in [t, t + \Delta t]$, we have $\delta(y_0, s + \Delta t) < \delta(y_0, s)$; it follows that there is no δ greater than $\delta_{t+\Delta t}^{\max}$ in the interval $[t + \Delta t, t + 2\Delta t]$. By extending this observation for every interval $[t + n \cdot \Delta t, t + (n + 1) \cdot \Delta t]$, with $n \in \mathbb{N}$, we can conclude that $\delta_{t+\Delta t}^{\max}$ weakly decreases with time.

Upper bound in space: The function $\delta_{t+\Delta t}^{\max}(y_0, t)$ provides an upper bound of δ for a given y_0. However, $\beta(d_0, t)$ depends on d_0 rather than directly on y_0. To address this, for any specified d_0, we must ensure that $\beta \geq \delta$ for every y_0 located at this particular distance from the equilibrium. The set of initial conditions y_0 fulfilling this condition can be defined as $\mathcal{Y}_0(d_0) = \{y_0 \in \mathcal{L} : ||y_g - y_0|| = d_0\}$. Considering this, we can introduce an upper bound dependent on d_0 by computing the maximum of each $\delta_{t+\Delta t}^{\max}(y_0, t)$, where $y_0 \in \mathcal{Y}_0$

$$\delta^{\max}(d_0, t) = \max_{\gamma_0 \in \mathcal{Y}_0(d_0)} (\delta^{\max}_{t+\Delta t}(\gamma_0, t)) \tag{13}$$

Strictly increasing/decreasing function: Similarly to $\delta^{\max}_{t+\Delta t}$, the function $\delta^{\max}$ also weakly decreases as a function of time (see Appendix A for details). This is not inherently problematic, as it verifies the properties of class- $\mathcal{KL}$ functions. However, β must strictly increase as a function of d_0, and for this to be the case, in our formulation, β must also strictly decrease as a function of time, $\forall d_0 \in \mathcal{L} \setminus \{\gamma_g\}$.

To achieve this, we consider β the evolution function of a first-order linear dynamical system with state z_t, using $\delta^{\max}$ as the reference. This leads to the equation

$$\dot{z}_t = \alpha(z_t - \delta^{\max}) \tag{14}$$

where $\alpha < 0$. With an initial condition z_0 greater than $\delta^{\max}_0 = \delta^{\max}(d_0, 0)$, we ensure that β strictly decreases with time. Moreover, β remains above $\delta^{\max}$, and consequently above δ, for all $d_0 \in \mathcal{L} \setminus \{\gamma_g\}$. Hence, by choosing $z_0 = \delta^{\max}_0 + d_0$, we ensure that β exhibits the desired increasing/decreasing properties. We can express β as

$$\beta(d_0, t) = z_0 + \int_0^t \dot{z}_s \mathrm{d}s. \tag{15}$$

Based on Prop. 1, Appendix A, we can confirm that our formulation for β is a valid class- $\mathcal{KL}$ upper bound of δ. This indicates that the first stability condition of Theorem 3 is satisfied, concluding our proof.

4.2.3. Loss Function

Crucially, the surrogate stability conditions from Theorem 4 can be enforced in a DNN by minimizing the following expression, $\forall \gamma_0 \in \mathcal{L}$ and $\forall t \in \mathbb{R}_{\geq 0}$

$$\max(0, m + d(\gamma_g, \gamma_{t+\Delta t}) - d(\gamma_g, \gamma_t)) \tag{16}$$

where $m \in \mathbb{R}_{>0}$. This function resembles the form of the triplet loss introduced in Section 3.4. However, in our context, γ_g serves as the anchor sample, $\gamma_{t+\Delta t}$ as the positive sample, and γ_t as the negative sample. In the following discussion, we explore how this expression induces the fulfillment of the surrogate stability conditions within a DNN.

Second Surrogate Condition: For any γ_0 where $x_0 \neq x_g$, minimizing (5) enforces transitions to progressively approach γ_g, as its minimization implies

$$d(\gamma_g, \gamma_{t+\Delta t}) + m \leq d(\gamma_g, \gamma_t) \tag{17}$$

Hence, $d(\gamma_g, \gamma_{t+\Delta t}) < d(\gamma_g, \gamma_t)$, that is, the second surrogate condition (see **Figure 5**).

First Surrogate Condition: In the case where $\gamma_0 = \gamma_g = \psi_\theta(x_g)$, that is, when the system is initialized at the equilibrium, enforcing a value of $\gamma_{t+\Delta t}$ different from γ_g would only increase the function in (5). Consequently, for $\gamma_0 = \gamma_g = \psi_\theta(x_g)$, (5) achieves its minimum when $\gamma_{t+\Delta t} = \gamma_g = \gamma_0$, equal to m. Therefore, this equation also enforces the first surrogate condition.

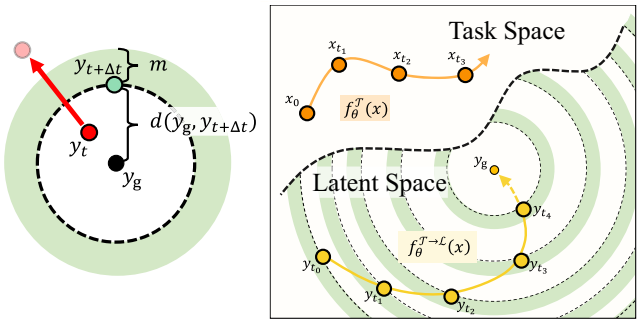


Figure 5. Left: Illustration of the effect of optimizing ℓ_{stable}. The red arrow depicts γ_t and $\gamma_{t+\Delta t}$ being modified to fulfill (6). Right: Example of trajectories generated with $f_\theta^{\mathcal{T}}$ and $f_\theta^{\mathcal{T}\to\mathcal{L}}$ post-training. Each $\gamma_{t+\Delta t}$ is closer to γ_g than its predecessor γ_t.

Finally, since $\delta(\gamma_0, t) = d(\gamma_g, \gamma_t)$, we present the stability loss function as

$$\ell_{\text{stable}} = \sum_{\gamma_0 \in \mathcal{B}^s} \sum_{t \in \mathcal{H}^s} \max(0, m + \delta(\gamma_0, t + \Delta t) - \delta(\gamma_0, t)) \tag{18}$$

In this equation, $\mathcal{B}^s$ denotes a batch of initial latent states γ_0. These states are derived by mapping initial states x_0 (sampled randomly from $\mathcal{T}$) via the function ψ_θ. Meanwhile, $\mathcal{H}^s$ represents a set of time instants t at which the loss is minimized. To compute this loss, we must recall that the evolution function of γ_t is represented as Φ^γ_θ. Consequently

$$\delta(\gamma_0, t) = ||\gamma_g - \Phi^\gamma_\theta(t, \gamma_0)|| \tag{19}$$

Given the absence of an analytical representation for Φ^γ_θ, we approximate it using the forward Euler method and optimize it using BPTT, in a similar manner to Section 4.1.

Importantly, optimizing this loss for all $t \in \mathbb{R}_{\geq 0}$ using BPTT is not feasible, as it would necessitate computing the loss over an infinite number of samples. Nevertheless, this limitation is not a significant concern when dealing with time-invariant dynamical systems. In such systems, any state γ_t can be equivalently represented by an initial condition γ_0, since both reside in the same state space $\mathcal{L}$. Consequently, when states are randomly sampled from $\mathcal{T}$, and thereby from $\mathcal{L}$, we are essentially sampling from the space of all possible γ_t.

4.3. On the Stability Loss Metric

Intentionally, we have not specified which distance function or metric should be used for computing the stability loss, since it should be selected depending on the geometry employed to describe the robot's state space.

To elaborate, recall that the learned dynamical system can be expressed as $\dot{x}_t = \phi_\theta(\gamma_t)$. It then follows that the output of our model is entirely determined by the latent state γ_t. This implies that if two different states x^a_t and x^b_t map to the same latent state, that is, $\gamma_t = \psi_\theta(x^a_t) = \psi_\theta(x^b_t)$, the time derivative computed by the learned dynamical system for both states will be identical. Such behavior would manifest in certain states if ψ_θ were a

nonbijective (More rigorously, the property being described is that of noninjective functions. However, for practical purposes, we can define the codomain of ψ_θ to be equal to its image, which is the property that an injective function must have to be bijective. Hence, in this specific context, we use these terms interchangeably) function. It would, therefore, be potentially harmful for the learning process to enforce ψ_θ to be nonbijective, as this would constrain the family of solutions to which the system can converge, hindering the DNN's optimization process. Ideally, we would like ψ_θ to have the capacity to converge to a bijective function if required, where each state x_t maps to a unique latent state y_t. Consequently, for any state x_t, it would be possible to compute a value of $\dot{x}_t$ that is independent of those calculated for any other state. **Figure 6** presents an example of bijective and nonbijective mappings between dynamical systems where the surrogate stability conditions are enforced.

It turns out that the capacity of ψ_θ to converge to a bijective function is closely linked to the metric used in calculating ℓ_{stable}, which should be chosen based on the geometry of the robot's state space. In the following subsection, we explore this relationship in more depth.

4.3.1. Homeomorphisms and State Space Geometry

To study under which conditions ψ_θ can converge to a bijective function, let us introduce the concept of topologically equivalent manifolds. Two manifolds, for example, $\mathcal{T}$ and $\mathcal{L}$, are topologically equivalent if a continuous and bijective mapping, known as an homeomorphism, exists between them.[45] In other words, two topologically equivalent manifolds can be stretched, compressed, or twisted into each other without tearing or gluing space. Since DNNs are continuous functions (We can assume this since every broadly used model is continuous.[46,47]), a bijective function ψ_θ would also serve as a homeomorphism (DNNs are commonly continuously differentiable, so sometimes in the literature the term *diffeomorphism* is employed instead, as diffeomorphisms are continuously differentiable homeomorphisms.) between $\mathcal{T}$ and $\mathcal{L}$.

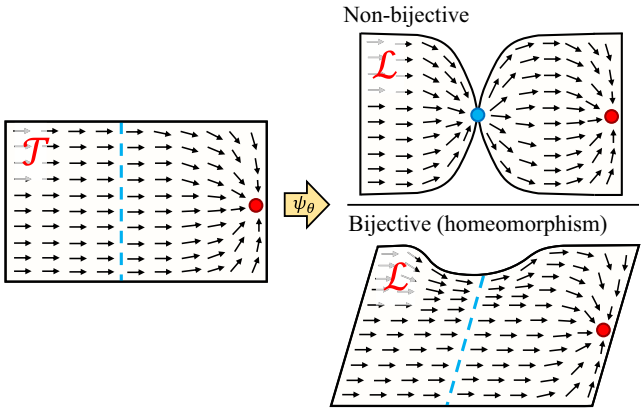


Figure 6. Nonbijective vs bijective solution. In the nonbijective case, every point on the blue line in $\mathcal{T}$ maps to the blue point in $\mathcal{L}$. In the bijective case, the original line undergoes a transformation that both compresses and skews it, yet a one-to-one mapping is maintained.

Moreover, since we have a notion of distance in both $\mathcal{T}$ and $\mathcal{L}$, it follows that these manifolds are metric spaces. A key property of metric spaces is that their topology is generated by their distance functions. Therefore, if two metric spaces are topologically equivalent, their metrics are termed as being equivalent.[48] It is important to clarify that this does not necessarily mean that the two distance functions are identical, but rather that they induce metric spaces that are homeomorphic to each other.

In our context, this implies that the distance function employed in the stability loss (7) must induce a topology in $\mathcal{L}$ that is equivalent to that of $\mathcal{T}$. For example, if orientations are described using unit quaternions, the topology of $\mathcal{T}$ would be spherical. Then, the stability loss metric should generate a topology that is homeomorphic to the sphere. Otherwise, it would be infeasible for the DNN to establish a homeomorphism between $\mathcal{T}$ and $\mathcal{L}$.

In this work, Section 5, we use unit quaternions to represent orientations in our robot experiments. For a detailed discussion on metrics and pose control in this context, the reader is referred to Appendix B.

4.4. Boundary Conditions

Lastly, it is crucial to ensure that a dynamical system evolving in $\mathcal{T}$ always remains within this manifold. Two scenarios are relevant to this work: 1) ensuring $\mathcal{T}$ is positively invariant with respect to $f_\theta^{\mathcal{T}}$ and 2) considering the state's geometry when computing the evolution of the dynamical system.

4.4.1. Positively Invariant Sets

In PUMA, the stability of a motion is enforced by randomly sampling points from $\mathcal{T}$ and minimizing ℓ_{stable}. Thus, stability cannot be ensured in regions where this loss is not minimized, that is, outside of $\mathcal{T}$. When boundaries are imposed on the robot's workspace, the learned dynamical system $f_\theta^{\mathcal{T}}$ can potentially evolve toward these boundaries, leaving $\mathcal{T}$. Hence, to ensure stability, a state evolving within $\mathcal{T}$ must not leave $\mathcal{T}$. In other words, $\mathcal{T}$ has to be a positively invariant set with respect to $f_\theta^{\mathcal{T}}$.[6,43]

To address this, we design the dynamical system so that it cannot leave $\mathcal{T}$ by construction. This can be achieved by projecting any transitions that would leave $\mathcal{T}$ back onto its boundary. For example, in Euclidean state spaces, $\mathcal{T}$ can be represented as a hypercube; therefore, in this case, this projection is achieved by saturating/clipping the points that leave $\mathcal{T}$. Furthermore, we found that introducing an additional loss, denoted as ℓ_∂, can be beneficial in enforcing this condition through the optimization process of the DNN. As proposed in ref. [6], this can be accomplished by using the scalar product between the dynamical system's velocity $v(x_t)$ (which equates to $f_\theta^{\mathcal{T}}$ for first-order systems) and the outward-pointing normal vector $n(x_t)$ at states within the boundary of $\mathcal{T}$. This product should be ensured to be equal to or less than zero. Then, to achieve this for DNNs, we introduce the following loss function

$$\ell_\partial = \max(0, n(x_t) \times v(x_t)) \tag{20}$$

4.4.2. Evolving in Non-Euclidean State Spaces

When modeling dynamical systems in non-Euclidean state spaces, it is crucial to ensure that states do not evolve outside the manifold representing them. For instance, unit quaternions must remain within the unit sphere. However, if these states are evolved using Euclidean geometry tools, such as the forward Euler integration method, deviations from the manifold are likely to occur.

Non-Euclidean state spaces are commonly defined within a higher-dimensional Euclidean space and can sometimes be constructed by incorporating constraints into this space. Consider the unit quaternion as an example: its state space consists of every vector in $\mathbb{R}^4$ with a unit norm, forming the 3-sphere $\mathcal{S}^3 \subseteq \mathbb{R}^4$. Therefore, in such scenarios, by integrating an operation that enforces these constraints, such as normalization for $\mathcal{S}^3$, into the structure of the DNN representing $f_\theta^{\mathcal{T}}$, we ensure that transitions remain within the manifold. Moreover, in this way, the distortions that this operation introduces into the dynamical system's output are factored into the DNN's optimization process, ensuring they are accounted for.

Alternatively, when we have access to the Riemannian metric of a state space manifold, it is possible to use the exponential and logarithmic maps to do Euclidean calculus in the tangent bundle of the manifold and then map the solution back on the manifold (the reader is referred to ref. [49] for more details).

5. Experimental Section

We validated our method with three datasets, each allowing us to study different aspects of it. For evaluation purposes, we used these datasets under the assumption of perfect tracking of the desired state derivatives, $\dot{x}_t^d$, provided by $f_\theta^{\mathcal{T}}$, without any involvement of robots in this process. Subsequently, we tested our method in two real-world settings using two different robots. We made our code implementation of PUMA publicly available at https://github.com/rperezdattari/Deep-Metric-IL-for-Stable-Motion-Primitives. Details on the DNN's hyperparameters optimization process are presented in Appendix C.

5.1. Euclidean Datasets

First, we evaluated our method using datasets of Euclidean motions. This approach allowed us to examine the performance of different variations of PUMA for Euclidean dynamical systems and compare their effectiveness against state-of-the-art methods.

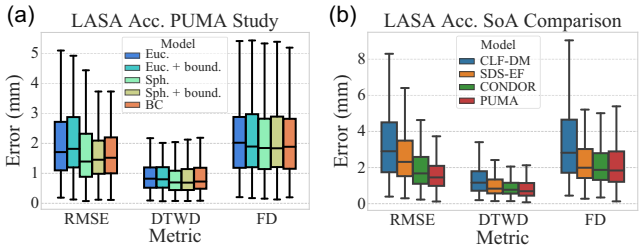
5.1.1. LASA

The LASA dataset[4] was composed of 30 human handwriting motions, each consisting of seven demonstrations of desired trajectories under different initial conditions. These demonstrations were 2D and designed to be modeled using first-order systems, that is, output desired velocities as a function of their positions. To compare accuracy performance between different models, we employed the same metrics in every experiment: 1) root mean squared error (RMSE), 2) dynamic time warping distance (DTWD),[50] and 3) Fréchet distance (FD).[51]

Accuracy: **Figure 7**a showcases the accuracy of four variations of PUMA. Here, we compared the performance of different distance metrics in ℓ_{stable} for motions in Euclidean spaces, focusing on the Euclidean distance and the great-circle (spherical) distance. Furthermore, we also examined the influence of ℓ_∂ on the accuracy of the learned motions. We used behavioral cloning (BC) without a stability loss as an upper performance bound for comparison. Interestingly, each variation of PUMA achieved a similar performance; however, PUMA with a spherical metric showed slightly better performance than PUMA with an Euclidean metric. This suggested that the DNN had no difficulties in mapping the Euclidean space $\mathcal{T}$ into a spherical space $\mathcal{S}$. Lastly, we could also observe that the use of ℓ_∂ did not harm the accuracy performance of PUMA.

Stability: The stability of the motions learned by PUMA hinged on the successful minimization of (4), which we needed to test after the learning process concluded empirically. To do this, we integrated the dynamical system over $\mathcal{L}$ time steps, starting from P initial states, and observed whether the system converges to the goal. The larger the P, the more accurate our results. If $\mathcal{L}$ was sufficiently large, the system should reach the goal after $\mathcal{L}$ steps. By measuring the distance between the last state visited and the goal, and confirming that it falls below a preset threshold ε, we could evaluate if a trajectory was successful (i.e., it converges to the goal).

Table 1 provides the stability results of BC and different PUMA variations. The data showed that every variation of PUMA successfully enforced stability across the dataset, generating no unsuccessful trajectories. Compared to BC, which had a 36.4653% rate of unsuccessful trajectories, the benefit of the proposed loss in enforcing stability became clear.



Comparison of different variations of PUMA.

Comparison of PUMA with other state-of-the-art (SoA) methods.

Figure 7. Accuracy study in the LASA dataset.

Table 1. Percentage of unsuccessful trajectories over the LASA dataset ($\mathcal{L} = 2500$, $P = 2500$, $\varepsilon = 1$ mm). Bold indicates the lowest values, which are four in this case.

Behavioral cloning	PUMA [Euc.]	PUMA [Euc. + ℓ_∂]	PUMA [Sph.]	PUMA [Sph. + ℓ_∂]
36.4653%	**0.0000%**	**0.0000%**	**0.0000%**	**0.0000%**

State-of-the-Art Comparison: Figure 7b presents an accuracy comparison of PUMA with other state-of-the-art methods, namely: 1) Control Lyapunov Function-based Dynamic Movements (CLF-DM) using GMR,[5] 2) Stable Dynamical System learning using Euclideanizing Flows (SDS-EF),[2] and 3) CONDOR. The results for PUMA corresponded to the best-performing variation in this dataset, that is, spherical distance with ℓ_∂. We observed that PUMA achieved competitive results, demonstrating similar performance in DTWD and FD to CONDOR and SDS-EF and slightly superior performance under RMSE.

Boundary Loss: **Figure 8** demonstrates the effect of the boundary loss ℓ_∂ in PUMA. Figure 8a,b presents an example of the qualitative performance of PUMA and CONDOR when no boundary loss is applied. In the top-left region of these images, PUMA exhibited nonsmooth trajectories at the boundary, which abruptly change direction due to the applied saturation (see Section 4.4). Conversely, CONDOR learned a smoother trajectory in this region. This feature in PUMA vanished when ℓ_∂ was applied, as depicted in Figure 8c. Finally, Figure 8d provides a quantitative evaluation of the boundary loss, confirming our qualitative observations.

In this work, this loss was only relevant for Euclidean state spaces because we do not introduce boundaries in the state space when controlling orientation in $\mathcal{S}^3$.

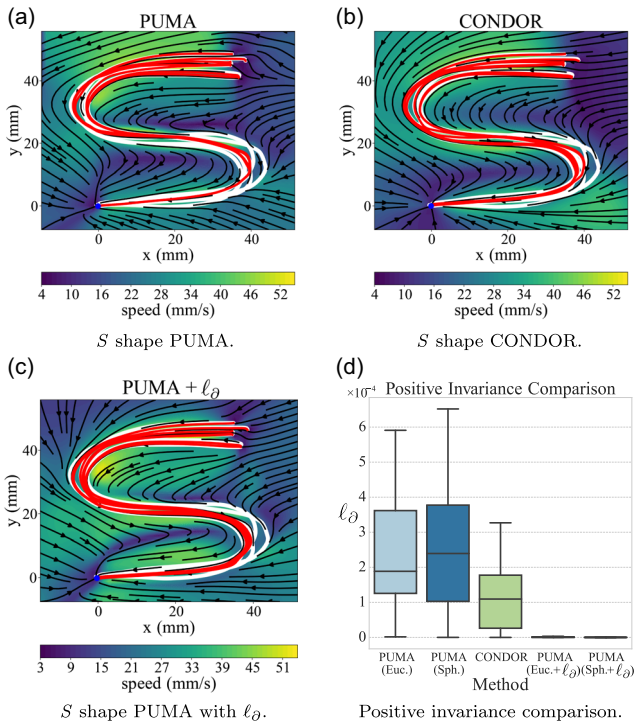


Figure 8. Positive invariance evaluation. a–c), white curves represent demonstrations. Red curves represent learned motions when starting from the same initial conditions as the demonstrations. The arrows indicate the vector field of the learned dynamical system. Figure d) provides a positive invariance comparison between CONDOR and the variations of PUMA.

5.1.2. LAIR

Contrary to the LASA dataset, the LAIR dataset[9] was specifically designed to evaluate second-order motions, that is, those that map current position and velocity to acceleration. This dataset comprised ten human handwriting motions, with the state being four-dimensional, encompassing a 2D position and velocity. The dataset's shapes contained multiple position intersections, intentionally designed to necessitate the use of at least second-order systems for their successful modeling.

CONDOR/PUMA Comparison: The state-of-the-art methods outlined in Section 5.1.1, excluding CONDOR, did not address the challenge of learning stable second-order systems. This can be attributed to the greater difficulty inherent in learning such systems compared to first-order systems. As a result, we compared the performance of PUMA with that of CONDOR. **Figure 9**a illustrates the comparative accuracy of both methods, with PUMA surpassing CONDOR on all metrics. PUMA's greater learning flexibility allowed it to converge to solutions beyond CONDOR's capabilities. Moreover, this enhanced flexibility, as depicted in Figure 9b, endowed PUMA with a more stable learning process. In practice, this made a significant difference, as motions with unsuccessful trajectories must be discarded when considering the system's stability. Therefore, greater learning stability results in a larger set of motions without unsuccessful trajectories, providing more alternatives of successfully learned systems to choose from.

Stability: **Table 2** presents the results of stability analysis for PUMA on the LAIR dataset. Similar to the findings with the LASA dataset, every variation of PUMA achieved a 0% rate of unsuccessful trajectories. This validated PUMA's ability to learn stable second-order motions.

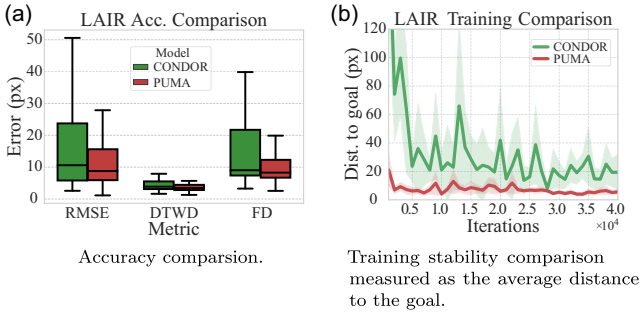


Figure 9. LAIR dataset PUMA/CONDOR comparison. a) Accuracy comparision, b) Training stability comparision measured as the average distance to the goal.

Table 2. Percentage of unsuccessful trajectories over the LAIR dataset ($\mathcal{L} = 2500$, $P = 2500$, $\varepsilon = 10$ px). Bold indicates the lowest values, which are four in this case.

Behavioral cloning	PUMA [Euc.]	PUMA [Euc. + ℓ_∂]	PUMA [Sph.]	PUMA [Euc. + ℓ_∂]
14.1160%	**0.0000%**	**0.0000%**	**0.0000%**	**0.0000%**

5.2. Non-Euclidean Dataset: LASA $\mathcal{S}^2$

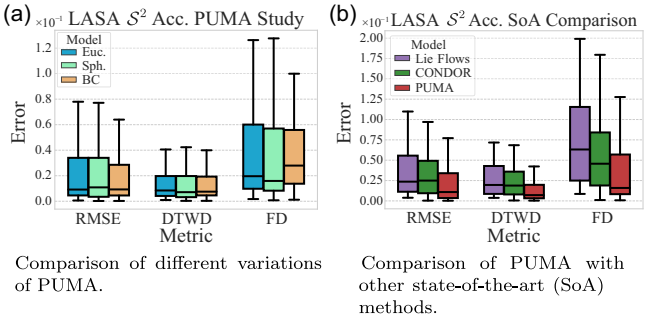
The LASA $\mathcal{S}^2$ dataset[11] comprised 24 motions, each with 3 demonstrations. Unlike the LASA dataset, the LASA $\mathcal{S}^2$ dataset represented these motions in spherical geometry, as indicated by its name, in $\mathcal{S}^2$. The sphere, where the motions evolved, was structured similarly to unit quaternions, though with one less dimension. Essentially, we had a 3D Euclidean space where vectors were constrained to have a unit norm. As a result, the state space was a 2D spherical manifold embedded in this 3D space. T his setup allowed us to examine the performance of PUMA in a manifold with similar attributes to those of unit quaternions. However, the more straightforward visualization of this setup facilitated a more intuitive analysis of PUMA's performance.

5.2.1. Accuracy

Figure 10a presents the performance of two variations of PUMA, namely when using Euclidean and spherical distance functions. Moreover, BC was included, serving as a lower-bound reference. As explained in Section 5.1.1, the boundary loss ℓ_∂ did not apply in this case and was, therefore, not evaluated. We could observe that the accuracy performance of both variations of PUMA was very similar, and BC performed slightly better than both of them.

5.2.2. Stability

Table 3 depicts the stability results for the LASA $\mathcal{S}^2$ dataset. In this case, we also conducted a stability analysis of CONDOR, which, as discussed in Section 2, should not be capable of ensuring stability in non-Euclidean state spaces. This assertion was confirmed by the data in Table 3, which showed that CONDOR yielded 7.3133% of unsuccessful trajectories. When compared to BC, which had a 15.9217% rate of unsuccessful trajectories, it can be inferred that CONDOR reduced the number of unsuccessful trajectories in non-Euclidean state spaces. However, its performance was still far from satisfactory. In contrast, both versions of PUMA achieved 0% of unsuccessful trajectories, marking a significant improvement. Moreover, this validated that the Euclidean distance was effective for enforcing stability in spheres, as it can induce spherical metrics, such as the chordal distance, in lower-dimensional manifolds (see Appendix B).



Comparison of different variations of PUMA.

Comparison of PUMA with other state-of-the-art (SoA) methods.

Figure 10. Accuracy study in LASA $\mathcal{S}^2$ dataset. a) Comparision of different variations of PUMA. b) Comparion of PUMA with other state-of-the art (SoA) methods.

Table 3. Percentage of unsuccessful trajectories over the LASA $\mathcal{S}^2$ dataset ($L = 2500$, $P = 2500$, $\varepsilon = 0.06$). Bold indicates the lowest values, which are four in this case.

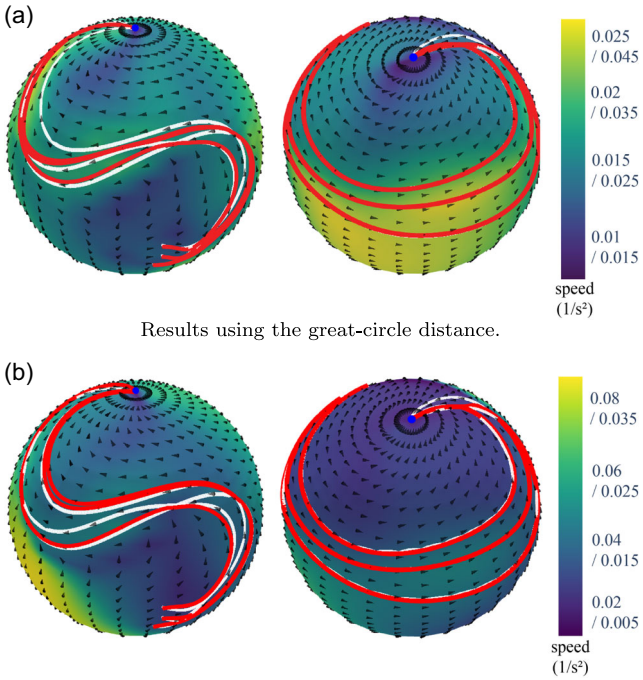
Behavioral cloning	CONDOR	PUMA [Euc.]	PUMA [Sph.]
15.9217%	7.3133%	**0.0000%**	**0.0000%**

5.2.3. Qualitative Analysis

Figure 11 illustrates motions learned with PUMA in $\mathcal{S}^2$ using both spherical and Euclidean distances. Within the observed region of the sphere, the vector field exhibited no spurious attractors. Furthermore, when initiated from the same conditions as the demonstrations, the trajectories in both cases showed an accurate reproduction of the motions.

5.2.4. State-of-the-Art Comparison

Figure 10b depicts a comparison between three methods: 1) CONDOR, 2) PUMA, and 3) Lie Flows,[11] a method developed for learning stable motions in non-Euclidean manifolds using tools from Lie Theory. PUMA outperforms both CONDOR and Lie Flows in terms of accuracy. Moreover, in contrast to CONDOR, PUMA also successfully ensured stability in this dataset.



Results using the great-circle distance.

Results using the Euclidean (chordal) distance.

Figure 11. Motions learned with PUMA in $\mathcal{S}^2$. White curves represent demonstrations. Red curves represent learned motions when starting from the same initial conditions as the demonstrations. The arrows indicate the vector field of the learned dynamical system. a) Results using the great-circle distance; b) results using the Euclidean distance.

5.3. Real-World Experiments

We validated our method in two real-world setups, using two different robots. Both robots had six degrees of freedom, with their end-effector pose represented in $\mathbb{R}^3 \times \mathcal{S}^3$, and were guided using PUMA. In these experiments, $f_\theta^{\mathcal{T}}$ was employed to generate velocity references in real time, which were then sent to the low-level controllers responsible for tracking these references in the robots. Note that throughout the experiments, we operated under the assumption that the target states of the robots were already known.

5.3.1. Greenhouse

This experiment was conducted in a greenhouse, where a robot was trained to reach a black marker on a tomato plant (see **Figure 12**). This marker, simulating a plant's peduncle, represented the target state the robot must reach to harvest the plant. Here, the marker allowed us to test the method without causing harm to the plant. This task presented significant challenges, as it required the robot to perform precise position and orientation control while considering the plant's complex geometry, which was nontrivial to model. In a practical setting, a motion library should be developed to account for the variability across different plants and targets. This library could then be used to select or combine suitable motions to create an appropriate movement strategy considering the plant's characteristics. This experiment represented a preliminary step toward achieving this objective.

We utilized an ABB IRB 1200 robot equipped with the Externally Guided Motion (Code: https://github.com/ros-industrial/abb_robot_driver.) module, allowing real-time joint velocity control. As the motion takes place in the end-effector space, the desired velocity at each time step was calculated using $f_\theta^{\mathcal{T}}$ and subsequently mapped to the joint space. This conversion was accomplished using the inverse kinematics module with joint limits from the Robotics Toolbox.[52] Similarly, we translated the robot's estimated joint state into the end-effector space using the forward kinematics module from the same toolbox. The same approach was taken to collect the demonstrations of this task directly in the end-effector space, where a space mouse was employed to teleoperate the robot.

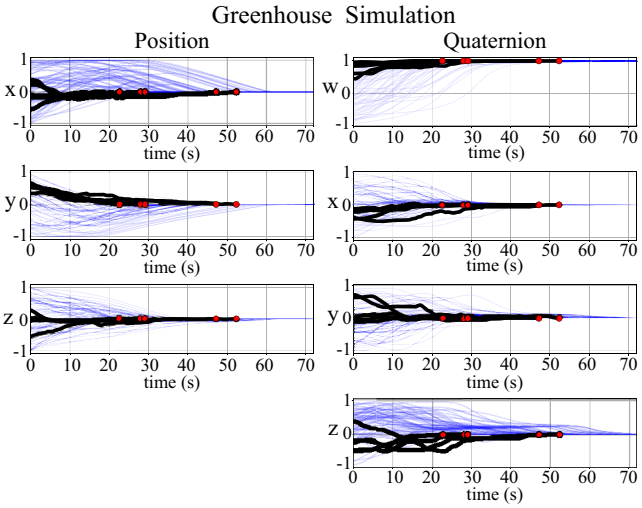


Figure 13. Simulated trajectories, as a function of time, of the greenhouse experiment. Blue trajectories correspond to evaluations of the model under different initial conditions and the black trajectory corresponds to the demonstration. The red point is the goal.

Figure 13 presents the demonstrations and simulations of motions conducted in this experiment. It can be observed that five demonstrations were provided. Note that these demonstrations do not have the same end time; only the final state needs to be consistent. Over a span of 70 s, all simulated trajectories visibly converge to the goal. For more details, the reader is referred to the attached video.

5.3.2. Hammer

The second experiment involved a robot accurately positioning a hammer next to other tools on a table. This task required precise control of the robot's position and orientation, as shown in Figure 12. We were interested in controlling the hammer's movement, so we assumed that the state of the hammer directly

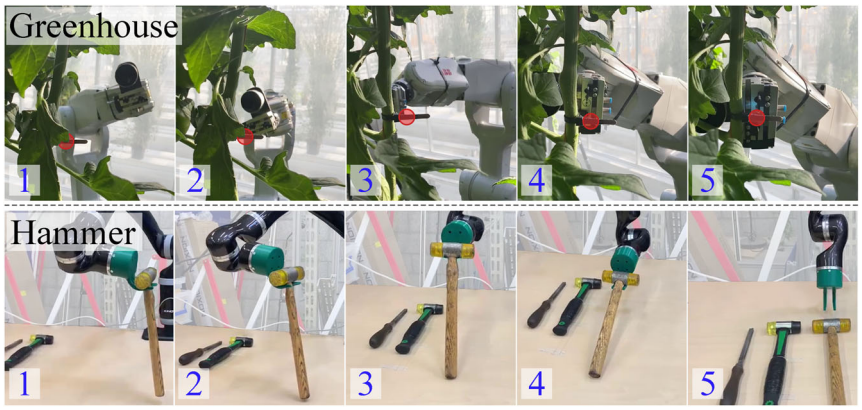


Figure 12. Two sequences of frames, each depicting a robot performing a task: 1) operating in a greenhouse and 2) placing a hammer. The red circle is used to highlight a target marker in the greenhouse experiment.

related to the state of the robot's end effector. This assumption was necessary because the employed low-level controller acted on the robot's end-effector state, and we ensured stability in this state space. Given that the hammer was attached to the robot through a double hook, this assumption was valid as long as the hammer's head was securely held. This assumption broke down toward the end of the motion when the robot placed the hammer on the table. However, since the final states of both the hammer and the robot were similar, we observed that ensuring stability in the end-effector's motion effectively guided the hammer's motion toward its goal state, as can be seen in Figure 12 and in the attached video.

We used the Kinova Gen2 Ultra lightweight robot arm with a double hook replacing its default gripper. The control commands obtained from $f_\theta^{\mathcal{T}}$ were directly sent to a Cartesian space controller from Kinova (Kinova's controllers: https://github.com/Kinovarobotics/kinova-ros). The demonstrations were collected through kinesthetic teaching, that is, physically moving the robot along desired paths. The simulated performance in this task was similar to that in the greenhouse, given that the state space was identical in both cases. We refer the reader to the attached video for examples of the robot executing this task from various initial conditions.

6. Conclusions

We introduced a novel approach for learning stable robotic motions in both Euclidean and non-Euclidean state spaces. To achieve this, we introduced a new loss function based on the triplet loss from the deep metric learning literature. We validated this loss both theoretically and experimentally.

Our approach, PUMA, showcased state-of-the-art performance in every experiment, both in terms of accuracy and stability. It was validated using datasets where the dynamical system evolution was simulated, as well as real robotic platforms where it was employed to provide control commands to low-level controllers.

Compared to previous work, PUMA offers not only an improvement in addressing non-Euclidean state spaces but also increased flexibility by reducing restrictions in the latent space of the DNN, leading to generally better performance. More specifically, in previous work, the latent dynamical system is constrained to evolve along straight lines toward the goal. In contrast, PUMA allows the latent dynamical system to converge toward a broader range of stable dynamical systems, since its loss function only enforces the latent dynamical system to reduce the distance toward the goal. This feature expands the set of feasible solutions available to the DNN during optimization, thereby enhancing the model's adaptability.

Lastly, while this paper's findings are promising, there are also limitations that can be addressed in future works. First, further exploration of the scalability properties of PUMA is needed, as the largest DNN input employed in this paper had seven dimensions. Second, we have so far focused on learning independent motion primitives for specific tasks. A relevant line of research would be integrating this model into a larger framework where multiple primitives are learned and combined together. Third, a topic that we did not address in this work is obstacle avoidance. Recent techniques, such as Geometric Fabrics,[53] exploit dynamical systems to represent motions that achieve full-body obstacle avoidance, which can be integrated with dynamical systems learned in the end-effector space of the robot. Consequently, an exciting avenue for future work is exploring the combination of these methods with PUMA. Finally, another interesting research direction is studying the integration of these approaches with the low-level control of the robots. Currently, it is assumed that the transitions requested by PUMA can be tracked by the controllers of the robot. However, this is not always the case, and it would be beneficial to incorporate this information into the learning framework.

Appendix A

Upper Bound β

In this section, we introduce and prove the proposition used in Section 4.2.2 to demonstrate that the surrogate stability conditions ensure asymptotic stability in the dynamical system $f_\theta^{\mathcal{T}}(x_t)$.

Proposition 1 (Existence of class-$\mathcal{KL}$ function). *Consider a dynamical system $\dot{y}_t = f(y_t)$ with $f : \mathcal{L}_{\mathbb{R}^n \to \mathbb{R}^n}$ continuously differentiable, and $\Phi_\theta^y(t, y_0) : \mathbb{R}_{\geq 0} \times \mathcal{L} \to \mathcal{L}$ as its evolution function. For a distance function d_t in $\mathcal{L}$, we define its evolution, for a given t and y_0, as $\delta : \mathcal{L} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, with $\delta(y_0, t) = ||y_g - \Phi_\theta^y(t, y_0)||$.*

Then, consider the function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defined as

$$\beta(d_0, t) = z_0 + \int_0^t \dot{z}_s ds \tag{A1}$$

with derivative

$$\dot{z}_t = \alpha(z_t - \delta^{max}(d_0, t)) \tag{A2}$$

where $\alpha \in \mathbb{R}_{<0}$ and

$$\delta^{\max}(d_0, t) = \max_{y_0 \in \mathcal{Y}_0} \left(\max_{s \in [t,\, t+\Delta t]} \delta(y_0, s) \right) \tag{A3}$$

with $\mathcal{Y}_0(d_0) = \{y_0 \in \mathcal{L} : ||y_g - y_0|| = d_0\}$ and $\Delta t \in \mathbb{R}_{>0}$. The initial condition z_0 is set as

$$z_0 = \delta_0^{max} + d_0 \tag{A4}$$

where $\delta_0^{\max} = \delta^{\max}(d_0, 0)$.

Then, under the conditions of Theo. 4, i.e., $\forall t \in \mathbb{R}_{\geq 0}$,

1) *$d_t = d_{t+\Delta t}$, for $y_0 = y_g$*
2) *$d_t > d_{t+\Delta t}$, $\forall y_0 \in \mathcal{L} \setminus \{y_g\}$*

there exists a class-$\mathcal{KL}$ function β such that $\forall y_t \in \mathcal{L}$ and $\forall t \in \mathbb{R}_{\geq 0}$,

$$||y_g - y_t|| \leq \beta(d_0, t) \tag{A5}$$

which can also be expressed as

$$\delta(y_0, t) \leq \beta(d_0, t) \tag{A6}$$

Proof. To prove that a function β is an upper bound of δ and is class-$\mathcal{KL}$, we need the following conditions to hold $\forall y_t \in \mathcal{L}$ and $\forall t \in \mathbb{R}_{\geq 0}$

i) $\beta(0,t)=0$,
ii) β decreases with t,
iii) $\delta \leq \beta$,
iv) β is continuous with respect to d_0 and t,
v) $\beta \to 0$ as $t \to \infty$,
vi) β strictly increases with d_0.

Therefore, we proceed to prove all of these conditions. A summary of this proof is depicted in **Figure A1**.

a) $\beta(0,t)=0$

To demonstrate that this condition is valid, we introduce Lemma 1. For this lemma to be applicable, a specific condition must be met. To validate this condition, we refer to Lemma 2 and Cor. 1, both of which will be introduced subsequently. We also present Cor. 2, which will prove beneficial in subsequent discussions.

Lemma 1 ($\beta(0,t)=0$ constant). *$\beta(0,t)=0$ if $\delta^{max}(0,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$.*

Proof. If $\delta^{\max}(0,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$, for $d_0=0$, from Equation (A4) and (A2), we can observe that $z_0=0$ and $\dot{z}_t=0$, $\forall t \in \mathbb{R}_{\geq 0}$. Replacing this in (8), we conclude that $\beta(0,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$.

Lemma 2 ($\delta=0$ constant). *$\delta(\gamma_g,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$, if:*
1) *$d_t = d_{t+\Delta t}$, for $\gamma_{t+\Delta t}=\gamma_g$* (Theo. 4, 1)
2) *$d_t > d_{t+\Delta t}$, $\forall \gamma_t \in \mathscr{L} \setminus \{\gamma_g\}$* (Theo. 4, 2)

Proof. Let us introduce t_g, which is the earliest $t \in \mathbb{R}_{\geq 0}$ where $\gamma_t=\gamma_g$. Then, condition 1 gives rise to two scenarios for every state visited after t_g :

i) *$\delta(\gamma_0,t)$ is constant:* A constant function satisfies $d_t = d_{t+\Delta t}$.

ii) *$\delta(\gamma_0,t)$ is periodic:* A set of functions could meet the condition $d_t = d_{t+\Delta t}$ by varying over time but always returning to the same value after Δt seconds. However, in this case, the only possibility is that, for all $t > t_g$, we have $\delta(\gamma_0,t)=\delta(\gamma_0,t+\Delta t)$, that is, a periodic function. This is because the time derivative of δ is solely defined by γ_t, since $\partial\delta/\partial t = \partial\delta/\partial\gamma_t \cdot \dot{\gamma}_t$, and both of these variables only depend on γ_t. Therefore, $\partial\delta/\partial t$ at γ_g, and at any state visited afterward, given the state, must always be the same.

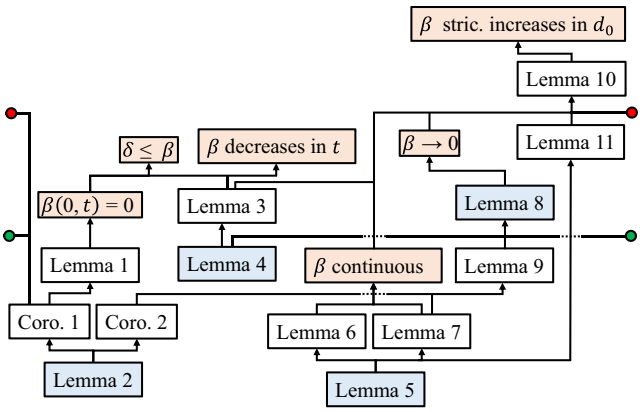


Figure A1. Summary of lemmas and corollaries related to the proof of Prop. 1. Bisque-colored boxes indicate the conditions necessary for the proposition to be true, while blue boxes represent lemmas relying directly on its assumptions and requirements.

This implies that δ must evolve over time to the same states as those to which it evolved Δt seconds ago, since for every state, the derivative is the same as it was Δt seconds before.

However, the periodic case for $\delta(\gamma_0,t)$ contradicts condition 2, which requires that δ strictly decreases after Δt seconds, and in the periodic scenario, it returns to the same value after Δt seconds. Hence, the only remaining scenario is that δ is constant after t_g, with a value of 0 (by definition). Lastly, by noting that $\gamma_0=\gamma_g$ implies that $t_g=0$, we get that $\delta(\gamma_g,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$.

Corollary 1 ($\delta^{\max}(0,t)=0$). *$\delta^{max}(0,t)=0$ if $\delta(\gamma_g,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$.*

Proof. If $\delta(\gamma_g,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$, and noting $d_0=0$ implies that every $\gamma_0 \in \mathcal{Y}_0$ in (10) is equal to γ_g, from (10), it follows that $\delta^{\max}(0,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$.

Corollary 2 ($\min(\delta^{\max})=0$). *$\min(\delta^{max})=0$ if $\delta^{max}(0,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$.*

Proof. Recall that δ is positive definite and that $\delta^{\max}$ takes the value of some δ. Then, $\delta^{\max}(0,t)=0$, $\forall t \in \mathbb{R}_{\geq 0}$, indicates that this is the minimum value $\delta^{\max}$ can achieve. Hence, $\min(\delta^{\max})=0$.

Then, provided that the conditions of Prop. 1 are met, the conditions of Lemma 2 are fulfilled. Therefore, $\delta(\gamma_g,t)=0$ for every $t \in \mathbb{R}_{\geq 0}$, and hence, $\delta^{\max}(0,t)=0$ (from Cor. 1). Consequently, we can employ Lemma 1 to demonstrate that $\beta(0,t)=0$ for all $t \in \mathbb{R}_{\geq 0}$.

b) β decreases with t, and $\delta \leq \beta$

We introduce Lemma 3, which presents one condition that, if satisfied, implies that $\beta > \delta$ and β strictly decreases over time, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$. After proving this lemma, we introduce Lemma 4, which is then used to show that this condition is satisfied in our case. Lastly, we combine this with the fact that $\beta(0,t)=0$ to conclude that β decreases with t (that is, is a non-increasing function), and $\delta \leq \beta$, $\forall d_0 \in \mathbb{R}_{\geq 0}$.

Lemma 3 ($\beta > \delta$ and strictly decreasing). *$\beta(d_0,t) > \delta(\gamma_0,t)$ and $\beta(d_0,t)$ strictly decreases with respect to t, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$, $\forall t \in \mathbb{R}_{\geq 0}$, if:*
1) *δ^{max} decreases with respect to t, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$.*

Proof. Although $\delta^{\max}$ evolves as a function of time, we can infer from (9) that, locally, β behaves as a linear first-order dynamical system, with the origin at $\delta^{\max}$. Given that $\alpha < 0$, it follows that β will converge toward $\delta^{\max}$ over time.

Moreover, (A4) indicates that $z_0 > \delta_0^{\max}$ for every $d_0 > 0$. Given the properties of linear first-order systems, β will strictly decrease toward $\delta_0^{\max}$, without ever reaching or overshooting this value. Combined with condition 1, which indicates that $\delta^{\max}$ decreases with respect to t for all $d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$, we can conclude that β will always be greater than $\delta^{\max}$ and will continue to strictly decrease toward this value as a function of time. Therefore, $\beta > \delta$ and β strictly decreases with respect to t, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$.

Lemma 4 ($\delta^{\max}$ decreases over time). *$\delta^{max}(d_0,t)$ decreases with respect to t, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$, $\forall t \in \mathbb{R}_{\geq 0}$, if:*
1) *$d_t > d_{t+\Delta t}$, $\forall t \in \mathbb{R}_{\geq 0}$, $\forall \gamma_0 \in \mathscr{L} \setminus \{\gamma_g\}$,* (Theo. 4, 2).

Proof. Let us define

$$\delta^{\max}_{t+\Delta t}(\gamma_0,t) = \max_{s \in [t,\, t+\Delta t]} \delta(\gamma_0,s) \tag{A7}$$

which allows us to write $\delta^{\max} = \max_{\gamma_0 \in \mathcal{Y}_0}(\delta^{\max}_{t+\Delta t})$. We will first prove that $\delta^{\max}_{t+\Delta t}$ decreases with t.

To do so, observe that condition 1 can be rewritten as $\delta(\gamma_0, t+\Delta t) < \delta(\gamma_0, t)$. Given that this condition holds true for every s in the interval $[t, t+\Delta t]$, none of the values of $\delta(\gamma_0, s)$ computed in this interval can exceed the maximum of this interval, that is, $\delta^{\max}_{t+\Delta t}(\gamma_0, t)$, after Δt seconds. If there were such a value $\delta(\gamma_0, s)$ that is greater than $\delta^{\max}_{t+\Delta t}(\gamma_0, t)$ after Δt seconds, it would imply that $\delta(\gamma_0, s)$ had increased, for some s, after Δt, contradicting condition 1. Therefore, $\delta^{\max}_{t+\Delta t}$ can only decrease as time progresses. Since this holds for all instances of t, we conclude that $\delta^{\max}_{t+\Delta t}$ decreases with respect to t, $\forall \gamma_0 \in \mathcal{L} \backslash \{\gamma_g\}$.

Now, the function $\delta^{\max}$ computes the maximum over a set of functions $\delta^{\max}_{t+\Delta t}$ with different initial conditions $\gamma_0 \in \mathcal{Y}_0(d_0)$. However, for any given d_0, $\forall \gamma_0 \in \mathcal{L} \backslash \{\gamma_g\}$, each one of these functions decreases with respect to time. Then, for any given interval of time, we have two possible scenarios: 1) $\delta^{\max}$ is equal to one function $\delta^{\max}_{t+\Delta t}$ (the current maximum). In this case, $\delta^{\max}$ decreases with time, since $\delta^{\max}_{t+\Delta t}$ decreases with time. 2) The function that is currently the maximum over the set of functions $\delta^{\max}_{t+\Delta t}$ reaches a point in time t' where it changes. At the point where the change occurs, the function will be lower than or equal to its previous values for $t < t'$, since the previous maximum is decreasing. Furthermore, it will be greater than or equal to the values for $t > t'$, since the new maximum also decreases.

Therefore, we can conclude that $\forall d_0$ where $Y_0 \in Y_0(d_0)$ is not γ_g, $\delta^{\max}$ decreases with respect to time. Moreover, by noting that $\forall Y_0 \in Y_0(d_0), \neq 0$ implies that $Y_0 \neq Y_g$, it follows that $\delta^{\max}$ decreases with respect to time, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$, $\forall t \in \mathbb{R}_{\geq 0}$.

As a consequence, the assumptions and conditions outlined in Prop. 1 enable us to apply Lemma 4 to demonstrate that the conditions of Lemma 3 are satisfied. This, in turn, leads us to the conclusion that $\beta > \delta$ and $\beta(d_0, t)$ strictly decreases with respect to t, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$, $\forall t \in \mathbb{R}_{\geq 0}$. Lastly, before we showed that, for $d_0 = 0$, $\beta(0, t) = 0$, $\forall t \in \mathbb{R}_{\geq 0}$. Therefore, in general, we have that $\beta \geq \delta$ and β decreases with respect to t, $\forall d_0 \in \mathbb{R}_{\geq 0}$, $\forall t \in \mathbb{R}_{\geq 0}$.

c) Continuity of β

We require β to be continuous with respect to t, for each fixed d_0, and with respect to d_0, for each fixed t. To achieve this, we will introduce Lemmas 5, 6, and 7.

Lemma 5. *[Continuity of δ] If $\dot{\gamma}_t = f(\gamma_t)$ is continuously differentiable, then $\delta(\gamma_0, t)$ is continuous with respect to γ_0 and t.*

Proof. Given that $\dot{\gamma}_t = f(\gamma_t)$ is continuously differentiable, it follows that $\Phi^{\gamma}_{\theta}(t, \gamma_0)$ is continuously differentiable with respect to both t and γ_0.[54] Moreover, since δ is a metric on $\mathcal{L}$ defined by $\delta = ||\gamma_g - \Phi^{\gamma}_{\theta}(t, \gamma_0)||$, it is continuous with respect to $\Phi^{\gamma}_{\theta}(t, \gamma_0)$.[55] Since $\Phi^{\gamma}_{\theta}(t, \gamma_0)$ is continuously differentiable and thus continuous, and the composition of two continuous functions is continuous, it follows that $\delta(\gamma_0, t)$ is continuous with respect to both γ_0 and t.

Lemma 6 ($\delta^{\max}$ continuous with respect to t). *δ^{max} is a continuous function with respect to t, if δ is continuous with respect to t.*

Proof. We will first prove this for $\delta^{\max}_{t+\Delta t}$ (as defined in (12)). Given that $\delta^{\max}_{t+\Delta t}$ is the maximum value of $\delta(\gamma_0, s)$ over the interval $[t, t+\Delta t]$, any change in $\delta^{\max}_{t+\Delta t}$ must be due to a change in δ for some s in the boundaries of $[t, t+\Delta t]$. Then, since δ changes continuously, $\delta^{\max}_{t+\Delta t}$ can only change continuously as well. Thus, $\delta^{\max}_{t+\Delta t}$ is continuous with respect to t.

However, we need to show that $\delta^{\max} = \max_{\gamma_0 \in \mathcal{Y}_0}(\delta^{\max}_{t+\Delta t})$ is continuous with respect to t. This follows from the fact that $\delta^{\max}$ computes the point-wise maximum along t over a set of continuous functions $\delta^{\max}_{t+\Delta t}$, which results in a continuous function.[56]

Lemma 7 ($\delta^{\max}$ continuous with respect to d_0). *δ^{max} is continuous with respect to d_0, if δ is continuous with respect to γ_0.*

Proof. Given the continuity of the function $\delta(\gamma_0, t)$ with respect to γ_0, the function $\delta^{\max}_{t+\Delta t}$ computes the point-wise maximum along γ_0 over a set of continuous functions, specifically $\{\delta(\gamma_0, s) : s \in [t, t+\Delta t]\}$. Consequently, $\delta^{\max}_{t+\Delta t}$ is also continuous in γ_0.[56]

Building on this, we seek to demonstrate that $\delta^{\max} = \max_{\gamma_0 \in \mathcal{Y}_0}(\delta^{\max}_{t+\Delta t})$ is continuous with respect to d_0. To accomplish this, we will use the maximum theorem.[57] This theorem states that if $\mathcal{Y}_0(d_0)$ is continuous with respect to d_0 and compact-valued (The concepts of compact-valued and continuous refer to those employed in the literature of set-valued functions/correspondences. For further details, we refer the reader to.[57]), and $\delta^{\max}_{t+\Delta t}(\gamma_0, t)$ is continuous in γ_0, then $\delta^{\max}(d_0, t)$ is continuous in d_0.

$\mathcal{Y}_0(d_0)$ is continuous because the relationship between each γ_0 and d_0, that is, the metric on $\mathcal{L}$, is continuous.[55] $\mathcal{Y}_0(d_0)$ is compact-valued if, for each d_0, $\mathcal{Y}_0$ constitutes a compact set. Given that $\mathcal{Y}_0 \mathbb{R}^n$, the *Heine-Borel theorem*[56] indicates that $\mathcal{Y}_0$ is compact if it is both closed and bounded. Every $\mathcal{Y}_0$ is closed as it fulfills an algebraic equation and can be contained within any ball of radius larger than d_0, making it bounded. Consequently, $\mathcal{Y}_0(d_0)$ is compact-valued.

Therefore, since $\delta^{\max}_{t+\Delta t}(\gamma_0, t)$ is continuous with respect to γ_0, and $\mathcal{Y}_0(d_0)$ is both continuous and compact-valued, the maximum theorem states that $\delta^{\max}(d_0, t)$ is continuous with respect to d_0.

Now, we can proceed to conclude about the continuity of β for both t and d_0.

Continuity in t: For any given d_0, since β evolves as a linear first-order system (as per (8)), its solution will exist provided that $\delta^{\max}$ is continuous with respect to t.[58] Then, under the assumptions of Prop. 1, Lemma 6 confirms that $\delta^{\max}$ is indeed continuous, provided that δ is continuous. From Lemma 5, we infer that δ is continuous with respect to t. Hence, β is well-defined for any given d_0, indicating that it is differentiable with respect to t and, therefore, continuous.

Continuity in d_0: For any given t, from (8), we know that β will be continuous with respect to d_0 as long as both of its terms, z_0 and the intregral of $\dot{z}_t$, are continuous. The continuity of both terms depends on the continuity of $\delta^{\max}$. Then, since Lemma 7 indicates that $\delta^{\max}$ is continuous with respect to d_0 provided that δ is continuous with respect to γ_0, and Lemma 5 confirms that this is indeed the case (given the assumptions of Prop. 1), we conclude that β is continuous with respect to d_0.

d) $\beta \to 0$ as $t \to \infty$

To prove that this condition holds, we will utilize Lemma 8. The application of this lemma necessitates the use of Lemma 4 and Cor. 1, which are already introduced previously, and Lemma 9, which is introduced afterward.

Lemma 8 (β converges to zero). *$\beta \to 0$ as $t \to \infty$ if:*
1) $\delta^{\max} \in [0, \max(\delta)]$,
2) δ^{max} *decreases with respect to t,* $\forall d_0 \in \mathbb{R}_{\geq 0}$, $\forall t \in \mathbb{R}_{\geq 0}$,
3) $d_t < d_{t-\Delta t}$, $\forall \gamma_0 \in \mathscr{L} \setminus \{\gamma_g\}$, $\forall t \in \mathbb{R}_{\geq 0}$ (Theo. 4, 2).
4) δ^{max} *surjective in* d_0,
5) $\delta^{max}(0, t) = 0$, $\forall t \in \mathbb{R}_{\geq 0}$.

Proof. According to (9), β approaches $\delta^{\max}$ as $t \to \infty$. Thus, demonstrating that $\delta^{\max} \to 0$ as $t \to \infty$ would imply that $\beta \to 0$ as $t \to \infty$.

Note that $\delta^{\max} \in [0, \max(\delta)]$ (condition 1) and $\delta^{\max}$ decreases for all t (condition 2). This indicates that as t goes to infinity, $\delta^{\max}$ must approach a limit $a \in [0, \max(\delta)]$. We will proceed to show that this limit can only be zero.

Let us define $\gamma_0^* \in \mathcal{Y}_0$ and $t^* \in \mathbb{R}_{\geq 0}$ as the variables in (A3) where the maxima are achieved for a given d_0 and t. Thus, $\delta^{\max} = \delta(\gamma_0^*, t^*)$. From condition 3, we know that $\delta(\gamma_0^*, t^* + \Delta t) < \delta(\gamma_0^*, t^*)$, $\forall \gamma_0^* \in \mathscr{L} \setminus \{\gamma_g\}$, $\forall t^* \in \mathbb{R}_{\geq 0}$. This leads to two scenarios.

γ_0^* remains constant: If γ_0^* does not change in the interval $[t, t^* + \Delta t]$, then $\delta(\gamma_0^*, t^*)$, and therefore, $\delta^{\max}$ must strictly decrease at some point within this interval. Otherwise, $\delta(\gamma_0^*, t^* + \Delta t) < \delta(\gamma_0^*, t^*)$ would not hold true.

γ_0^* changes: If γ_0^* changes during the interval $[t, t^* + \Delta t]$, given that $\delta^{\max}$ decreases with t (condition 2), this change can only occur if $\delta(\gamma_0^*, t^*)$, and hence, $\delta^{\max}$ strictly decreases at some point within the interval.

In either case, $\delta^{\max}$ strictly decreases at some point in the interval $[t, t^* + \Delta t]$, $\forall d_0 \in \mathbb{R}_{\geq 0} \setminus \{0\}$, $\forall t \in \mathbb{R}_{\geq 0}$.

Taking this into account, if the limit was some value $a \neq 0$, a contradiction would arise. If $a \neq 0$, we would always have a $d_0 \neq 0$ such that $\delta^{\max}(d_0, 0) = a$ (conditions 4 and 5). This implies that $\delta^{\max}$ must become lower than a before $t^* + \Delta t$, and, since it is decreasing (condition 2), it will remain lower as time progresses. Therefore, the only feasible value for the limit is $a = 0$, which confirms $\delta^{\max} \to 0$ as $t \to \infty$. Consequently, $\beta \to 0$ as $t \to \infty$.

Lemma 9 ($\delta^{\max}$ with d_0). *$\delta^{max} \in [0, \max(\delta)]$, and δ^{max} surjective with respect to d_0, if:*
1) $\min(\delta^{max}) = 0$,
2) δ^{max} *continuous with respect to* d_0.

Proof. Given that $\delta^{\max}$ computes a maximum over values of δ, we get $\max(\delta^{\max}) = \max(\delta)$. Then, we know that $\min(\delta^{\max}) = 0$ (condition 1) and that $\delta^{\max}$ is continuous with respect to d_0 (condition 2). Therefore, as a consequence of the intermediate value theorem, $\delta^{\max}$ can take any value between $\min(\delta^{\max})$ and $\max(\delta^{\max})$, that is, $\delta^{\max} \in [0, \max(\delta)]$. Moreover, each value of $\delta^{\max} \in [0, \max(\delta)]$ must have at least one corresponding value of d_0; hence, in this interval, $\delta^{\max}(d_0, t)$ is surjective with respect to d_0.

To summarize, given the conditions and assumptions of Prop. 1, Lemmas 4 and 9 and Cor. 1 indicate that the conditions of Lemma 8 are fulfilled, implying that $\beta \to 0$ as $t \to \infty$.

e) β strictly increases with d_0

To prove this, we will introduce Lemma 10, which needs a condition supported by Lemma 11, introduced afterward.

Lemma 10 (β strictly increases with d_0) *β strictly increases with d_0 if:*
1) z_0 *strictly increases with* d_0,
2) $\delta^{max}(0, t) = 0$, $\forall t \in \mathbb{R}_{\geq 0}$,
3) β *strictly decreases with t,* $\forall d_0 \in \mathbb{R}_{\geq 0}$,
4) β *continuous with t,*
5) $\beta \to 0$ *as* $t \to \infty$.

Proof. For a fixed t, the β strictly increases in d_0 if for any two numbers d_0^a and d_0^b (where $d_0^a < d_0^b$) within its domain, the condition $d_0^a < d_0^b$ implies $\beta^a(d_0^a, t) < \beta^b(d_0^b, t)$.

By condition 1, z_0 strictly increases with d_0, so we have $z_0^a(d_0^a) < z_0^b(d_0^b)$. Moreover, since (any) z_0 is nonnegative (see (A4)), it follows that $z_0^b > z_0^a \geq 0$. Given condition 2, this indicates that $d_0^b > 0$.

Recalling (A1), we have

$$\beta^b = z_0^b + \int_0^t \dot{z}_s ds \tag{A8}$$

Since $d_0^b > 0$, it follows from condition 3 that $\beta(d_0^b, t)$ strictly decreases with respect to t (i.e., $\dot{z}_t < 0$). Furthermore, β is continuous with t and approaches zero as time goes to infinity (conditions 4 and 5). This, combined with $z_0^b > z_0^a$, implies that there must exist a time $t^a > 0$ such that

$$\beta^b = \underbrace{z_0^b + \int_0^{t^a} \dot{z}_s ds}_{z_0^a} + \int_{t^a}^{t} \dot{z}_s ds. \tag{A9}$$

Since $\dot{z}_t < 0$, starting from the same initial condition, a larger integration interval results in a smaller value of β. Therefore, we deduce that $\beta^b > \beta^a$, as β^a follows the same format as Equation (13) (when replacing the terms equivalent to z_0^a, with z_0^a) but with an integral that starts at 0 instead of t^a, and $0 < t^a$. Thus, under the given conditions, β strictly increases with respect to d_0.

Lemma 11 (z_0 strictly increases with d_0). *z_0 strictly increases with d_0 if:*
1. $\delta(\gamma_0, t)$ continuous with respect to t.

Proof. For z_0 to strictly increase with d_0, for any two numbers d_0^a and d_0^b within its domain, the inequality $d_0^a < d_0^b$ must imply $z_0^a(d_0^a) < z_0^b(d_0^b)$. From (11), we have $z_0 = \delta_0^{\max} + d_0$. Given that d_0 strictly increases with itself, it suffices to analyze the behavior of $\delta_0^{\max}$.

Recall that $\delta_0^{\max} = \max_{\gamma_0 \in \mathcal{Y}_0}(\delta_{t+\Delta t}^{\max}(\gamma_0, 0))$ and that 0pt plustw500pt plus-tw $\delta_{t+\Delta t}^{\max}(\gamma_0, 0)$ computes the maximum over the set $\{\delta(\gamma_0, s) : s \in [0, \Delta t]\}$. We will refer to this set as $\mathcal{W}(\gamma_0)$. Importantly, $\mathcal{W}(\gamma_0)$ contains $\delta(\gamma_0, 0) = d_0$ for all γ_0. Then, for any d_0^a and d_0^b satisfying $d_0^a < d_0^b$, the behavior of $\delta_0^{\max}$ can be studied in three different scenarios. Let $\gamma_0^a \in \mathcal{Y}_0(d_0^a)$ be the state that maximizes $\delta_{t+\Delta t}^{\max}$ for d_0^a, then

i) $d_0^a = \delta_{t+\Delta t}^{\max}(\gamma_0^a, 0)$: This scenario occurs when d_0^a is the maximum of $\mathcal{W}(\gamma_0^a)$. Then, if $\gamma_0^b \in \mathcal{Y}_0(d_0^b)$ is the state that maximizes $\delta_{t+\Delta t}^{\max}$ for d_0^b, and hence, $d_0^b \in \mathcal{W}(\gamma_0^b)$, the maximum of $\mathcal{W}(\gamma_0^b)$ cannot be lower than d_0^b. Consequently, since $d_0^a < d_0^b$, we get $\delta_{t+\Delta t}^{\max}(\gamma_0^a, 0) < \delta_{t+\Delta t}^{\max}(\gamma_0^b, 0)$. As both γ_0^a and γ_0^b are those that maximize $\delta_{t+\Delta t}^{\max}$ over γ_0, for d_0^a and d_0^b, respectively, we conclude that $\delta_0^{\max}(d_0^a) < \delta_0^{\max}(d_0^b)$.

ii) $d_0^a < \delta_{t+\Delta t}^{\max}(\gamma_0^a, 0) \leq d_0^b$: This scenario occurs when the maximum of $\mathcal{W}(\gamma_0^a)$ is not d_0^a, and d_0^b is greater than or equal to this maximum. Following a similar reasoning to the above, we can conclude that $\delta_0^{\max}(d_0^a) \leq \delta_0^{\max}(d_0^b)$.

iii) $d_0^a < d_0^b < \delta_{t+\Delta t}^{\max}(\gamma_0^a, 0)$: This scenario occurs when the maximum of $\mathcal{W}(\gamma_0^a)$ is not d_0^a, and d_0^b is lower than this maximum. Provided that δ is continuous with t (condition 1), in the set $\mathcal{W}(\gamma_0^a)$, since, for $t = 0$, $\delta = d_0^a$, and for some $t^* > 0$, $\delta = \delta_{t+\Delta t}^{\max}(\gamma_0^a, 0)$, then we know there must exist a t^b, with $0 < t^b < t^* \leq \Delta t$, where $\delta = d_0^b$.

Now, observe that $t^* \in [t^b, t^b + \Delta t]$. Therefore, $\delta_{t+\Delta t}^{\max}(\gamma_0^a, t^b)$, which computes the maximum over $\{\delta(\gamma_0^a, s) : s \in [t^b, t^b + \Delta t]\}$, cannot be lower than $\delta_{t+\Delta t}^{\max}(\gamma_0^a, 0)$. Moreover, we can select γ_t^a at time t^b, that is, $\gamma_{t^b}^a$, as a new initial condition such that $\delta_{t+\Delta t}^{\max}(\gamma_{t^b}^a, 0) = \delta_{t+\Delta t}^{\max}(\gamma_0^a, t^b)$. Since $\delta(\gamma_{t^b}^a, 0) = d_0^b$, we get that $\gamma_{t^b}^a \in \mathcal{Y}_0(d_0^b)$. Hence, even though $\gamma_{t^b}^a$ might not maximize $\delta_{t+\Delta t}^{\max}$ for $\mathcal{Y}_0(d_0^b)$, we know that the maximum, achived at γ_0^b, must be at least greater than or equal to $\delta_{t+\Delta t}^{\max}(\gamma_{t^b}^a, 0)$.

Consequently, we have that $\delta_{t+\Delta t}^{\max}(\gamma_0^b, 0) \geq \delta_{t+\Delta t}^{\max}(\gamma_{t^b}^a, 0)$, and $\delta_{t+\Delta t}^{\max}(\gamma_{t^b}^a, 0) = \delta_{t+\Delta t}^{\max}(\gamma_t^a, t^b) \geq \delta_{t+\Delta t}^{\max}(\gamma_0^a, 0)$. Hence, $\delta_{t+\Delta t}^{\max}(\gamma_0^a, 0) \leq \delta_{t+\Delta t}^{\max}(\gamma_0^b, 0)$, and, therefore, $\delta_0^{\max}(d_0^a) \leq \delta_0^{\max}(d_0^b)$.

In summary, for any d_0^a and d_0^b with $d_0^a < d_0^b$, it is always true that $\delta_0^{\max}(d_0^a) \leq \delta_0^{\max}(d_0^b)$, meaning $\delta_0^{\max}$ is nondecreasing with respect to d_0. As the sum of a nondecreasing function ($\delta_0^{\max}$) and a strictly increasing function (d_0) results in a strictly increasing function, we conclude that z_0 strictly increases with d_0.

From Lemma 10, we can conclude that β strictly increases with respect to d_0, since assuming the conditions of Prop. 1, and that we have proven that β is continuous in t, Lemmas 11, 3, and 8 indicate that the conditions of Lemma 10 are fulfilled.

To summarize, we have demonstrated that all the requirements for $\beta(d_0, t)$ to be a class-$\mathcal{KL}$ function, and to serve as an upper bound for $\delta(\gamma_0, t)$, are met for every $\gamma_t \in \mathscr{L}$ and for all $t \in \mathbb{R}_{\geq 0}$. With this, our proof of Prop. 1 is complete.

Appendix B

B Learning on Spherical Manifolds

In this work, we employ unit quaternions to control orientation; therefore, we consider two distance functions that are relevant when learning spherical geometries: 1) great-circle distance and 2) chordal distance.

Great-Circle Distance: This is the distance along a *great circle*. A great circle is the largest circle that can be drawn on any given sphere, defining the shortest distance between two points. In the context of unit quaternions, which have a unitary norm, this distance is equivalent to the central angle α subtended by two points on the sphere. Hence, we can define it as

$$d_{\text{g.c.}} = \alpha. \tag{B1}$$

Chordal Distance: This is a distance that can be computed when an n-sphere is embedded in a higher-dimensional Euclidean space, that is, $\mathcal{S}_{\mathbb{R}^{n+1}}^n$. Then, by computing the Euclidean distance in $\mathbb{R}^{n+1}$ between two points in $\mathcal{S}^n$, we induce a distance in $\mathcal{S}^n$, corresponding to the chordal distance.[59–61] As the name suggests, this distance is the length of the chord connecting two points in an n-sphere. Hence, it is defined as

$$d_{\text{chord}} = 2r \sin(\alpha/2) \tag{B2}$$

where r is the radius of the n-sphere.

These distances are depicted in **Figure B1**. Since both define the same topology $\mathcal{S}^n$, they are considered to be equivalent and can be utilized in PUMA to enforce stability when states are represented as unit quaternions. The great circle distance is especially suited to spherical spaces as it defines the shortest path, or the geodesic, between two points. Conversely, the chordal distance is straightforward to apply because it naturally arises when calculating the Euclidean distance at the output of ψ_θ. This is due to $\mathscr{L}_{\mathbb{R}^m}$, where $m > n$ represents the output size of ψ_θ.

B1 Comments on Local Stability

Considering a 1D sphere, it is notable that at $\alpha = \pi$ (with respect to the goal), both the great-circle distance and the chordal distance can decrease in two possible ways: by evolving either to the right or to the left (see Figure B1). Consequently, to ensure these distances decrease in the region around $\alpha = \pi$, one of these options should be selected. However, this would require the dynamical system to instantly change its direction at this point, rendering $f_\theta^{\mathcal{T}}$ discontinuous. Yet, in Theorem 4, we assume this function is continuous, as continuity is a prerequisite for the class-$\mathcal{KL}$ upper bound β to also be continuous, which is necessary to prove stability.

Therefore, by extending this idea to higher-dimensional spheres, this implies that when employing these metrics, we can only enforce local asymptotic stability for $\mathcal{S}^n \setminus \{p\}$, where p is the point at $\alpha = \pi$. Moreover, due to the continuity of $f_\theta^{\mathcal{T}}$, this point must correspond to a zero and, therefore, represents an unstable equilibrium. This property is not a flaw in the great-circle distance or the chordal distance. Rather, it is a result of the topology of $\mathcal{S}^n$, which does not allow for the existence of a single stable equilibrium. This is a consequence of the Poincaré–Hopf theorem.[62]

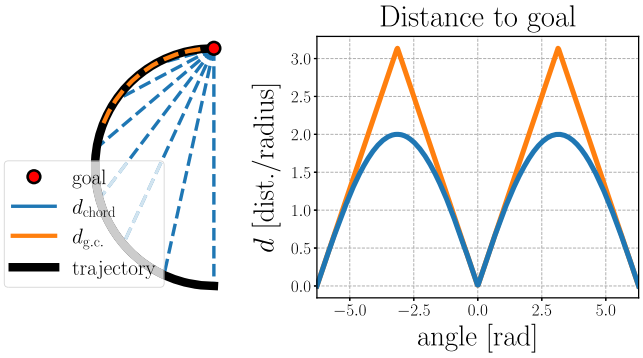


Figure B1. Left: A spherical trajectory toward a goal at the north pole. The chordal and great-circle distances at specific points are indicated as d_{chord} and $d_{\text{g.c.}}$, respectively. Right: Distances as a function of the central angle between points.

B2 Pose Control

Until now, our discussion has centered on the applicability of our method for unit quaternions. However, practical control of a robot's pose requires simultaneous control of both its position (Euclidean) and orientation (non-Euclidean). This can be achieved by using a product metric, that is, a metric resulting from the Cartesian product of spaces. In this case, we are looking at the product $\mathbb{R}^3 \times \mathcal{S}^3$.

A simple product metric is the sum of the metrics from each space in the product,[48] for example, $d_{\mathbb{R}^n} + d_{\mathcal{S}^n}$ for $\mathbb{R}^n \times \mathcal{S}^n$, where $d_{\mathbb{R}^n}$ is a distance in $\mathbb{R}^n$ and $d_{\mathcal{S}^n}$ is a distance in $\mathcal{S}^n$. However, it is not straightforward to do this in $\mathscr{L}$ without modifying the DNN structure. This is because the latent states y_t are an entangled representation of the robot states x_t, making it impossible to simply add together the distances of the Euclidean and non-Euclidean parts in the latent space.

Interestingly, when the product metric is computed using manifolds of identical topology, such as $\mathbb{R} \times \mathbb{R}$, the resulting metric is equivalent to the one obtained by directly computing the metric in the higher-dimensional space ($\mathbb{R}^2$ in this example).[48] Hence, for such scenarios, there is no need to explicitly disentangle the states in $\mathscr{L}$; instead, we can compute the metric directly in the complete latent space. Our case, nevertheless, is more complex since the topologies of $\mathcal{S}^3$ and $\mathbb{R}^3$ are different. Fortunately, we found two ways of easily overcoming this limitation.

B2.1 Pose in Euclidean Space

We have observed that an Euclidean metric, the chordal distance, can generate spherical metrics in lower-dimensional manifolds. This becomes particularly relevant when a Euclidean metric is employed in $\mathbb{R}^m$, where m exceeds the dimensionality of our state space (6 in our case). In this scenario, the metric is equivalent to $d_{\mathbb{R}^3} + d_{\mathbb{R}^{m'}}$ for $m' > 3$ and $3 + m' = m$. Given that $\mathcal{S}^3$ can be induced inside $\mathbb{R}^{m'}$, this suggests that $\mathbb{R}^3 \times \mathcal{S}^3$ can be induced in $\mathbb{R}^m$ when using the Euclidean distance in this space.

B2.2 Pose in Spherical Space: Finally, we also note that the great-circle distance can be employed to achieve the same objective. Suppose we compute this distance in $\mathcal{S}^6$. Then, it would be equivalent to $d_{\mathcal{S}^3} + d_{\mathcal{S}^3}$. Interestingly, a diffeomorphism can be found between $\mathbb{R}^n$ and a subset of $\mathcal{S}^n$, for example, the stereographic projection.[63] This implies that it is feasible to use the metric $d_{\mathcal{S}^3}$ in $\mathscr{L}$ and find a valid representation for $\mathbb{R}^3$ within a subspace of $\mathcal{S}^3$. Consequently, the product space $\mathbb{R}^3 \times \mathcal{S}^3$ can be represented within a subset of $\mathcal{S}^6$.

Appendix C

C Hyperparameter Optimization

To optimize the hyperparameters of the different variations of PUMA employed in the LASA and LAIR datasets, we utilized the Tree Parzen Estimator.[64] This optimization method builds a probability model that facilitates the selection of the most promising set of hyperparameters in each optimization round. For the implementation of the Tree Parzen Estimator, we used the Optuna API.[65]

To evaluate the selected sets of hyperparameters, we employ a loss function $\mathscr{L}_{\text{hyper}}$ composed of two components. The first component, $\mathscr{L}_{\text{acc}}$, assesses the accuracy of the trained model. This is done by computing the RMSE between the demonstrations and the trajectories simulated by the learned model, in a manner similar to the approach used in the experiments section. The second component, $\mathscr{L}_{\text{goal}}$, quantifies the average distance between the final points of trajectories, simulated using the learned model, and the target goal. Thus, we have

$$\mathscr{L}_{\text{hyper}} = \mathscr{L}_{\text{acc}} + \gamma \mathscr{L}_{\text{goal}} \tag{C1}$$

Table C1. Hyperparameter optimization results of the different variations of PUMA.

Hyperparameter	Opt.?	Hand-tuned value/initial opt. guess				Optimized value		
		Euc.	Sph.	Euc. (2nd order)	Sph. (2nd order)	Sph.	Euc. (2nd order)	Sph. (2nd order)
Stability loss margin [m]	✓	1.250e−8	1.250e−4	1.250e−4	1.250e−4	3.012e−05	2.424e−08	2.919e−7
Triplet imitation loss weight [λ]	✓	1	1	1	1	3.496	1.022e−1	4.473e−1
Window size imitation [$\mathscr{H}^i$]	✓	14	14	14	14	13	14	14
Window size stability [$\mathscr{H}^s$]	✓	4	1	1	1	13	11	11
Batch size imitation [$\mathscr{B}^i$]	✗	250	250	250	250	–	–	–
Batch size stability [$\mathscr{B}^s$]	✗	250	250	250	250	–	–	–
Neural network								
Optimizer	✗	Adam	Adam	Adam	Adam	–	–	–
Number of iterations	✗	40000	40000	40000	40000	–	–	–
Learning rate	✓	1e−4	1e−4	1e−4	1e−4	8.574e−4	1.670e−4	1.245e−4
Activation function	✗	GELU	GELU	GELU	GELU	–	–	–
Num. layers [ψ_θ, ϕ_θ]	✗	(3, 3)	(3, 3)	(3, 3)	(3, 3)	–	–	–
Neurons/hidden layer	✗	300	300	300	300	–	–	–
Layer normalization	✗	Yes	Yes	Yes	Yes	–	–	–

where γ is a weighting factor. After initial tests, we settled on $\gamma = 3.5$.

To make the computationally intensive process of optimizing hyperparameters more feasible, we employed six strategies: 1) reducing the overhead of the objective function, 2) limiting the size of the evaluation set, 3) utilizing Bayesian optimization, 4) applying pruning techniques, 5) selecting a subset of hyperparameters for optimization, and 6) employing an optimization range. For details on the first five strategies, the reader is referred to.[9] The sixth strategy consists of restricting the hyperparameter search to a predefined range.

Table C1 presents the results of this optimization process. We can observe the hyperparameters that were optimized, their initial values (chosen based on preliminary tests), and their final optimized values. The ranges for hyperparameter optimization are as follows: 1) $m: [1e-9, 1e-1]$, 2) $\lambda: [1e-1, 10]$, 3) $\mathscr{H}^i: [1, 14]$, 4) $\mathscr{H}^s: [1, 14]$, and 5) learning rate: $[1e-5, 1e-3]$. Note that the variations using the boundary loss are not included in the table, since in those cases the same hyperparameters were employed and the boundary loss was added with a weight of 0.001. The same holds for the behavioral cloning case. Lastly, regarding Section 4.4.2 it is important to note that in every experiment involving spherical state spaces, the dynamical system was kept within the manifold by normalizing the forward Euler integration output.

Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

Acknowledgements

The authors would like to thank Xin Wang, Thomas Versmissen, Bastiaan Vroegindeweij, Rekha Raja, Gert Kootstra, and Eldert van Henten of the Agricultural Biosystems Engineering Group at Wageningen University and Research. Their assistance during the real-world experiments of this work, conducted at their facilities, was invaluable.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Rodrigo Pérez Dattari: Conceptualization: (lead); Formal analysis: (lead); Investigation: (lead); Methodology: (lead); Software: (lead); Validation: (lead); Visualization: (lead); Writing—original draft: (lead); Writing—review and editing: (lead). **Cosimo Della Santina**: Formal analysis: (supporting); Writing—review and editing: (supporting). **Jens Kober**: Conceptualization: (supporting); Formal analysis: (supporting); Supervision: (lead); Writing—review and editing: (supporting).

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Keywords

deep metric learning, deep neural networks, dynamical systems, imitation learning, motion primitives, non-Euclidean geometries

Received: February 22, 2024
Revised: August 11, 2024
Published online:

[1] N. Perrin, P. Schlehuber-Caissier, *Syst. Control Lett.* **2016**, *96*, 51.
[2] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, N. Ratliff, *2nd Annual Conf. on Learning for Dynamics and Control (L4DC)*, **2020**.
[3] J. Urain, M. Ginesi, D. Tateo, J. Peters, *2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2020**, pp. 5231–5237.
[4] S. M. Khansari-Zadeh, A. Billard, *IEEE Trans. Robot.* **2011**, *27*, 943.
[5] *Robot. Auton. Syst.* **2014**, *62*, 752.
[6] A. Lemme, K. Neumann, R. F. Reinhart, J. J. Steil, *Neurocomputing* **2014**, *141*, 3.
[7] N. Figueroa, A. Billard, *Int. J. Robot. Res.* **2022**, *41*, 312.
[8] A. Ude, B. Nemec, T. Petrić, J. Morimoto, *2014 IEEE International Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2014**, pp. 2997–3004.
[9] R. Pérez-Dattari, J. Kober, *IEEE Trans. Robot.* **2023**, *39*, 3909.
[10] J. Zhang, H. B. Mohammadi, L. Rozo, *Conf. on Robot Learning*, PMLR **2022**.
[11] J. Urain, D. Tateo, J. Peters, *IEEE Robot. Autom. Lett.* **2022**, *7*, 12569.
[12] M. Saveriano, F. J. Abu-Dakka, V. Kyrki, *Robot. Auton. Syst.* **2023**, *169*, 104510.
[13] F. Schroff, D. Kalenichenko, J. Philbin, *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, IEEE, Piscataway, NJ **2015**, pp. 815–823.
[14] M. Kaya, H. S. Bilge, *Symmetry* **2019**, *11*, 1066.
[15] S. Calinon, A. Pistillo, D. G. Caldwell, *2011 IEEE/RSJ International Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2011**, pp. 3413–3418.
[16] H. Ravichandar, A. S. Polydoros, S. Chernova, A. Billard, *Annu. Rev. Control Robot. Auton. Syst.* **2020**, *3*, 297.
[17] A. Paraschos, C. Daniel, J. R. Peters, G. Neumann, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc. **2013**, *26*.
[18] M. Y. Seker, M. Imre, J. H. Piater, E. Ugur, *Robotics: Science and Systems* **2019**, *10*.
[19] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, *Neural Comput.* **2013**, *25*, 328.
[20] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, G. Neumann, *IEEE Robot. Autom. Lett.* **2023**, *8*, 2325.
[21] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, J. Peters, *2014 IEEE International Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2014**, pp. 2831–2837.
[22] A. Pervez, Y. Mao, D. Lee, *2017 IEEE-RAS 17th International Conf. on Humanoid Robotics (Humanoids)*, IEEE, Piscataway, NJ **2017**, pp. 191–197.
[23] R. Pahič, B. Ridge, A. Gams, J. Morimoto, A. Ude, *Neural Netw.* **2020**, *127*, 121.
[24] S. Bahl, M. Mukadam, A. Gupta, D. Pathak, *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 5058.
[25] N. Figueroa, A. Billard, *Conf. on Robot Learning*, PMLR **2018**, pp. 927–946.
[26] H. Ravichandar, I. Salehi, A. Dani, *Conf. on Robot Learning* , PMLR **2017**, pp. 369–378.
[27] K. Neumann, J. J. Steil, *Robot. Auton. Syst.* **2015**, *70*, 1.

[28] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, C. Xu, *IEEE Trans. Syst. Man Cybernet. Syst.* **2017**, *49*, 1175.
[29] H. Ravichandar, A. Dani, *Dynamic Systems and Control Conf.*, Vol. *57250*, American Society of Mechanical Engineers, New York, NJ **2015**, p. V002T27A008.
[30] C. Blocher, M. Saveriano, D. Lee, *2017 14th International Conf. on Ubiquitous Robots and Ambient Intelligence (URAI)*, IEEE, Piscataway, NJ **2017**, pp. 124–129.
[31] W. Zhi, T. Lai, L. Ott, F. Ramos, in *L4DC*, PMLR **2022**, pp. 508–519.
[32] P. Pastor, L. Righetti, M. Kalakrishnan, S. Schaal, *2011 IEEE/RSJ International Conf. on Intelligent Robots and Systems*, IEEE, Piscataway, NJ **2011**, pp. 365–371.
[33] L. Koutras, Z. Doulgeri, *Conf. on Robot Learning*, PMLR **2020**, pp. 293–302.
[34] M. Lang, S. Hirche, *IEEE Robot. Autom. Lett.* **2017**, *2*, 1601.
[35] M. Arduengo, A. Colomé, J. Lobo-Prat, L. Sentis, C. Torras, *J. Ambient Intell. Humaniz. Comput.* **2023**, 1.
[36] J. Silvério, L. Rozo, S. Calinon, D. G. Caldwell, *2015 IEEE/RSJ international Conf. on Intelligent Robots and Systems (IROS)*, IEEE, Piscataway, NJ **2015**, pp. 464–470.
[37] I. Havoutis, S. Calinon, *Auton. Robots* **2019**, *43*, 713.
[38] M. J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, D. G. Caldwell, *IEEE Robot. Autom. Lett.* **2017**, *2*, 1240.
[39] S. Kim, R. Haschke, H. Ritter, *Robot. Auton. Syst.* **2017**, *87*, 28.
[40] Y. Huang, F. J. Abu-Dakka, J. Silvério, D. G. Caldwell, *IEEE Trans. Robot.* **2020**, *37*, 82.
[41] F. J. Abu-Dakka, Y. Huang, J. Silvério, V. Kyrki, *Robot. Auton. Syst.* **2021**, *141*, 103761.
[42] H. C. Ravichandar, A. Dani, *Auton. Robots* **2019**, *43*, 897.
[43] H. K. Khalil, in *Nonlinear Systems*, 3rd ed., Prentice Hall, Upper Saddle River, NJ **2002**.
[44] C. M. Kellett, *Math. Control, Signal. Syst.* **2014**, *26*, 339.
[45] T. Needham, in *Visual Differential Geometry and Forms: A Mathematical Drama in Five Acts*, Princeton University Press, Princeton, NJ **2021**.
[46] I. Goodfellow, Y. Bengio, A. Courville, in *Deep Learning*, MIT Press, Cambridge, Mass **2016**.
[47] A. Géron, in *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly Media, Inc., Sebastopol, CA **2022**.
[48] E. Deza, M. M. Deza, M. M. Deza, E. Deza, in *Encyclopedia of Distances*, Springer, New York **2009**.
[49] S. Sommer, T. Fletcher, X. Pennec, in *Riemannian Geometric Statistics in Medical Image Analysis*, Elsevier, Amsterdam **2020**, pp. 3–37.
[50] M. Müller, in *Information Retrieval for Music and Motion*, Springer Berlin, Heidelberg, Berlin, Heidelberg **2007**, p. 69.
[51] T. Eiter, H. Mannila, Computing Discrete Fréchet Distance, Technische Universitat Wien, Tech. Rep. CD-TR 94/64, **1994**.
[52] P. Corke, J. Haviland, *2021 IEEE International Conf. on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ **2021**, pp. 11 357–11 363.
[53] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, B. Boots, N.D. Ratliff, *IEEE Robot. Autom. Lett.* **2022**, *7*, 3202.
[54] V. I. Arnold, in *Ordinary Differential Equations*, 3rd ed., Springer Science & Business Media, Berlin/Heidelberg, Germany **1992**, pp. 97–98.
[55] J. R. Munkres, in *Topology*, 2nd ed., Pearson Education, London, England **2000**, pp. 119–126.
[56] C. C. Pugh, in *Real Mathematical Analysis*, 2nd ed., Undergraduate Texts in Mathematics, Springer, New York **2015**, pp. 81, 97.
[57] K. C. Border, in *Fixed Point Theorems with Applications to Economics and Game Theory*, Cambridge University Press, Cambridge, England **1985**, pp. 53–66.
[58] R. K. Nagle, E. B. Saff, A. D. Snider, in *Fundamentals of Differential Equations*, 9th ed., Pearson, London, England **2018**, p. 53.
[59] J. M. Lee, in *Riemannian Manifolds: An Introduction to Curvature*, Vol. *176*, Springer Science & Business Media, Berlin/Heidelberg, Germany **2006**.
[60] C. Berg, in *Complex Analysis*, Matematisk Afdeling, Københavns Universitet, København, Denmark **2008**.
[61] J. Jeong, M. Jun, M. G. Genton, *Stat. Sci.* **2017**, *32*, 501.
[62] V. Guillemin, A. Pollack, in *Differential Topology*, Vol. *370*, American Mathematical Society, Providence, RI **2010**.
[63] J. Oprea, in *Differential Geometry and Its Applications. MAA* The Mathematical Association of America, Washington, DC **2007**.
[64] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, *Advances in Neural Information Processing Systems*, Curran Associates, Inc. **2011**, *24*.
[65] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, *Proc. of the 25th ACM SIGKDD International Conf. on Knowledge Discovery & Data Mining*, **2019**, pp. 2623–2631.