Safe and stable motion primitives via imitation learning and geometric fabrics

Saray Bakker*, Rodrigo Pérez-Dattari*, Cosimo Della Santina*, Wendelin Böhmer† and Javier Alonso-Mora*
* Department of Cognitive Robotics, † Department of Software Technology, Delft University of Technology
Email: S.Bakker-7@tudelft.nl

***Abstract*—Using the language of dynamical systems, Imitation learning (IL) provides an intuitive and effective way of teaching stable task-space motions to robots with goal convergence. Yet, these techniques are affected by serious limitations when it comes to ensuring safety and fulfillment of physical constraints. With this work, we propose to solve this challenge via TamedPUMA, an IL algorithm augmented with a recent development in motion planning called geometric fabrics. We explore two variations of this approach, which we name the forcing policy method and the compatible potential method. Making these combinations possible requires two enabling factors: the possibility of learning second-order dynamical systems by imitation and the availability of a potential function that is compatible with the learned dynamics. In this paper, we show how these conditions can be met when using an IL strategy called PUMA. The result is a stable imitation learning strategy within which we can seamlessly blend geometrical constraints like collision avoidance and joint limits. Beyond providing a theoretical analysis, we demonstrate TamedPUMA with simulated and real-world tasks, including a 7-degree-of-freedom manipulator that is trained to pick a tomato from a crate in the presence of obstacles.**

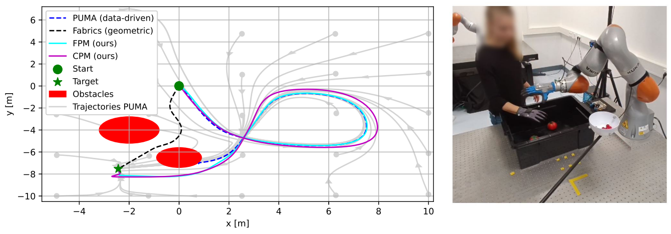


Fig. 1: On the left, we show trajectories of the proposed Forcing Policy Method (FPM) and Compatible Potential Method (CPM) against the baselines (vanilla geometric fabrics and vanilla PUMA) on a 2D point-mass example in the presence of obstacles.

I. Introduction

As robotic solutions rapidly enter unstructured sectors such as agriculture, household operations, and the food industry, there is a critical need for efficient methods to quickly train robots for new tasks. These sectors demand that robots safely interact with dynamic, fragile environments where humans are present. Currently, experts manually program these tasks, a method that is costly and not scalable for widespread use.

A possible solution to this societal challenge comes from Imitation Learning (IL). Using this technique robots can learn motion profiles from demonstrations provided by non-expert users. Furthermore, by encoding the learned trajectories as solutions of a dynamical system, established mathematical tools from dynamical system theory can be used to guarantee global convergence to the goal, as we discuss more in detail in Sec. I-A. In a robotics context, these learned dynamical systems encode the navigation policy towards a goal within a task space - such as the evolution of the end-effector pose of a manipulator while pouring water in a glass from all possible initial locations. However, this focus on task space motions renders IL fundamentally limited when it comes to considering physical constraints involving the body of the robot impacting with itself or interacting with the external environment. Substantial recent research has looked into the problem but, as we discuss in Sec. I-A, to the best of our knowledge there exists no solution that can simultaneously ensure stability and real-time fulfillment of physical constraints for systems with many degrees of freedom.

This work introduces TamedPUMA, a safe and stable extension of the IL algorithm PUMA [17]. The key enabling idea behind this new method is to look at the learned stable motion primitives as the navigation policy within a recently introduced geometrical framework for motion planning called geometric fabrics [23]. To make this possible, the learned model has to be formulated as a 2nd order (neural) dynamical system as fabrics operate within the Finsler Geometry framework where vector fields must be defined at the acceleration level [1]. Also, the learned dynamics must admit an *aligned* potential, which roughly means a scalar function whose gradient is aligned with the acceleration field when the velocity is null. In the paper, we show how we can ensure that both conditions are met. We propose two variations on TamedPUMA: the forcing policy method, treating the learned deep neural network (DNN) as a forcing term within the geometric fabrics formulation, and the compatible potential method, introducing an energy-regulation term requiring the design of a so-called compatible potential. For both, we analyze the end-to-end performance both analytically and experimentally (Fig. 1).

A. Related work

Learning stable dynamical systems from demonstrations Multiple methodologies have been introduced for learning stable dynamical systems [2, 6]. An influential methodology in learning stable dynamical systems is Dynamical Movement Primitives (DMPs) [7] ensuring convergence towards a simple manually-designed dynamical system. This approach is extended to non-Euclidean state spaces [26], probabilistic environments [15], and in the context of DNN [19, 24]. To ensure stability, several approaches enforce a specific structure on the function approximators, such as enforcing positive or negative definiteness [10, 14, 5], or invertibility [18, 20, 27]. This restricts the learned motion profile and limits the scalability of these methods to large and complex function approximators. In contrast, [16, 17] have showcased better scalability properties by using tools from deep metric learning [9] to enforce stability via a contrastive loss. Importantly, [17] extends the ideas of [16] to more general scenarios, achieving better results in non-Euclidean state spaces and second-order dynamical systems, which are necessary for integrating such frameworks with fabrics.

Geometric motion planning Geometric motion planning techniques are based on the classical technique of Operation Space Control [11], achieving stable and converging behavior for kinematically redundant robots using differential geometry [3]. Recently, Riemannian Motion Policies and Geometric Fabrics have been introduced, where the importance metric and forcing term can be decoupled [22, 13, 21]. For geometric fabrics, convergence is thereby guaranteed with simple construction rules [23, 29], and the approach is extended to dynamic environments [25]. These geometric policies are shown to be beneficial in designing human-like motions [12] and have a high planning frequency compared to optimization-based methods [25].

Combining geometric fabrics and learning Geometric fabrics have been considered in conjunction with reinforcement learning [28] and Bayesian learning [8]. Most interestingly for the present work, [30, 31] have looked into combining fabrics with IL. The method they propose is essentially different from TamedPUMA, as they directly learn the fabric, which results in limitations in terms of motion expressiveness.

II. Preliminaries

In this section, we introduce fundamental concepts for trajectory generation using artificial dynamical systems.

A. Geometric fabrics

A dynamical system describes the behavior of a system using differential equations. In *geometric fabrics* [23, 21], these dynamical systems describe an artificial system generating desired trajectories for a robotic system. The desired motions are described using second-order nonlinear time-invariant dynamical systems in multiple task spaces $\mathcal{X}_j$, which are mapped to and combined in $\mathcal{C}$-space. A task variable $\boldsymbol{x}_j \in \mathcal{X}_j$ denotes the value of the state representation for the j-th task space, where $j \in [M]$, M denotes the number of task spaces, and $[M] = \{j \in \mathbb{N} : j \leq M\}$. The relation between the joint state $\boldsymbol{q}$ in configuration space and a given task space is stated via a twice-differential map $\phi_j : \mathcal{C} \to \mathcal{X}_j$. The desired dynamical systems consist of two parts: one that is energy-conserving $\tilde{\boldsymbol{h}}$ and another that is energy-decreasing $\boldsymbol{f}$:

$$\ddot{\boldsymbol{q}} = \tilde{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{1}$$

The energy-conservative part takes care of all avoidance tasks, e.g., joint limit avoidance and obstacle avoidance, while the energy-decreasing part drives the system towards the goal.

Each avoidance task can be described by energy-conservative fabrics, $\ddot{\boldsymbol{x}}_j = \tilde{\boldsymbol{h}}(\boldsymbol{x}_j, \dot{\boldsymbol{x}}_j)$, also denoted as a *spec* $\mathcal{S} = (\tilde{\boldsymbol{M}}_j, \tilde{\boldsymbol{\xi}}_j)_{\mathcal{X}_j}$, within its own task space $\mathcal{X}_j$ which conserves a Finsler energy. To construct a whole-body policy for the robotic system, the task-space fabrics are mapped to the configuration space using a *pullback operation* and summed. The pullback operation $\text{pull}_{\phi_j} : \mathcal{X}_j \to \mathcal{C}$ is constructed as a function of ϕ_j, and maps the energy-conserving fabric to the configuration space, $\tilde{\boldsymbol{h}}_j : \mathcal{S}_j = (\tilde{\boldsymbol{M}}_j, \tilde{\boldsymbol{\xi}}_j)_{\mathcal{C}}$. The specs in configuration space are summed, e.g. $(\mathcal{S}_1 + \mathcal{S}_2)_{\mathcal{C}} = (\tilde{\boldsymbol{M}}_1 + \tilde{\boldsymbol{M}}_2, \tilde{\boldsymbol{\xi}}_1 + \tilde{\boldsymbol{\xi}}_2)_{\mathcal{C}}$ where the resulting dynamical system $\ddot{\boldsymbol{q}} = \tilde{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is a fabric as well, since the summation and pullback operations are closed under algebra. This combined fabric can be forced towards the minimum of a potential $\boldsymbol{\psi}$ and damped with a positive definite damping matrix β (energy-decreasing term).

B. Learning stable motion primitives via PUMA

Policy via neUral Metric leArning (PUMA) represents the dynamical system $\boldsymbol{f}_\theta^{\mathcal{T}}$ in one of the task spaces $\mathcal{X}_j$ (commonly the robot's end effector's space), denoted as $\mathcal{T}$, as a DNN with weights θ. The weights are optimized to imitate a set of demonstrations while ensuring convergence to a goal state $\boldsymbol{x}_\text{g} \in \mathcal{T}$ [17, 16]. In other words, $\boldsymbol{x}_\text{g}$ must be a globally asymptotically stable equilibrium in the region of interest.

To enforce stability, a specialized loss is introduced and optimized alongside an imitation loss. To design this loss, it is necessary first to define a latent space $\mathcal{L}$ as the output of a hidden layer l of the DNN, such that

$$\ddot{\boldsymbol{x}} = \boldsymbol{f}_\theta^{\mathcal{T}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{\varphi}_\theta(\boldsymbol{\rho}_\theta(\boldsymbol{x}, \dot{\boldsymbol{x}})), \tag{2}$$

where $\boldsymbol{\rho}_\theta : \mathcal{X} \to \mathcal{L}$ encodes the first $1, ..., l$ layers and $\boldsymbol{\varphi}_\theta : \mathcal{L} \to \mathcal{X}$ the last $l+1, ..., L$ layers. Then, a *projected system* $\boldsymbol{f}_\theta^{\mathcal{T}\to\mathcal{L}}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ can be defined in $\mathcal{L}$ as

$$\ddot{\boldsymbol{y}} = \boldsymbol{f}_\theta^{\mathcal{T}\to\mathcal{L}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{\partial \boldsymbol{\rho}_\theta(\boldsymbol{x}, \dot{\boldsymbol{x}})}{\partial t}. \tag{3}$$

These definitions enable the formulation of the *stability conditions* introduced in [16], which are used to construct the following stability loss in [17]:

$$\underbrace{\sum_{\boldsymbol{y}_0 \in \mathcal{B}} \sum_{t \in \mathcal{H}} \max(0, m + d(\boldsymbol{y}_\text{g}, \boldsymbol{y}(\boldsymbol{y}_0, t + \Delta t)) - d(\boldsymbol{y}_\text{g}, \boldsymbol{y}(\boldsymbol{y}_0, t)))}_{\ell_\text{stable}}, \tag{4}$$

where $d(\cdot, \cdot)$ is a distance function, m is a small margin hyperparameter, $\boldsymbol{y}(\boldsymbol{y}_0, t)$ represents the value of $\boldsymbol{y}$ for an initial condition $\boldsymbol{y}_0$ and time t. Here, $\mathcal{B}$ is a batch of initial conditions obtained at every iteration by randomly sampling the workspace, and $\mathcal{H}$ contains different values of t up to some time horizon T. To imitate the set of demonstrations, this method minimized the combined loss $\ell_\text{PUMA} = \ell_\text{IL} + \lambda \ell_\text{stable}$, where ℓ_IL is the behavioral cloning loss introduced in [16] and λ is a weight factor.

III. TamedPUMA: Combining learned stable motion primitives and fabrics

With learned stable motion primitives, complex tasks can be learned from demonstrations, while converging to the goal. By incorporating these learned dynamical systems into the navigation policy of geometric fabrics, stable and safe motions are generated respecting whole-body obstacle avoidance and physical constraints of the robot. In the following subsection, we introduce two variations of our method TamedPUMA.

A. The Forcing Policy Method (FPM)

First, we introduce the Forcing Policy Method (FPM). For this purpose, we define the dynamical system $\boldsymbol{f}_\theta^{\mathcal{C}}$ in configuration space resulting from applying a *pullback operation*, a map from task to configuration space, to the learned system, via PUMA, in $\mathcal{T}$:

$$\ddot{\boldsymbol{q}} = \boldsymbol{f}_\theta^{\mathcal{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \text{pull}_{\phi^{\mathcal{T}}}\left(\boldsymbol{f}_\theta^{\mathcal{T}}(\boldsymbol{x}, \dot{\boldsymbol{x}})\right). \tag{5}$$

Then, leveraging the definition of a forced system from Eq. (1), in the FPM we propose to use the pulled system obtained via PUMA as the forcing policy,

$$\ddot{\boldsymbol{q}} = \tilde{\boldsymbol{h}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{f}_\theta^{\mathcal{C}}(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{6}$$

Assuming ℓ_{PUMA} has already been minimized, the system $\boldsymbol{f}_\theta^{\mathcal{T}}$ comes to rest at $\boldsymbol{x}_{\text{g}}$, implying that $\boldsymbol{f}_\theta^{\mathcal{C}}$ converges to $\boldsymbol{q}_{\text{g}}$ where multiple values of $\boldsymbol{q}_{\text{g}}$ may exist in the case of a redundant system. This collection of states $\boldsymbol{q}_{\text{g}}$ corresponds to the *zero set* of $\boldsymbol{f}_\theta^{\mathcal{C}}$. From Proposition II.17 in [21], we know that if the system in Eq. (6) reaches the zero set of $\boldsymbol{f}_\theta^{\mathcal{C}}$, it will stay there (which comes from the observation that fabrics are conservative). However, although unlikely in practice, this system can still come to rest in a local minimum. The criteria that the system should converge to $\boldsymbol{q}_{\text{g}}$, is therefore not always met. In Sec. III-B, we propose a method with stronger convergence guarantees.

B. The Compatible Potential Method (CPM)

As a second approach, we propose the Compatible Potential Method (CPM) that exploits the concept of *compatible potentials* to obtain a stronger notion of convergence. A potential compatible with a dynamical system generally points in the same direction as the system's vector field. More formally:

Definition 1 (Compatible potential [21]). *A potential function ψ is compatible with $\boldsymbol{f}$ if: (1) $\partial\psi(\boldsymbol{q}) = \boldsymbol{0}$ if and only if $\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{0}) = \boldsymbol{0}$, and (2) $-\partial\psi^\top \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{0}) > \boldsymbol{0}$ wherever $\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{0}) \neq \boldsymbol{0}$.*

From this, [21] introduces Theorem III.5, which states that given a dynamical system with a compatible potential, then the system

$$\ddot{\boldsymbol{q}} = \text{energize}_{\mathcal{H}}[\boldsymbol{h} + \boldsymbol{f}] + \gamma(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{7}$$

converges to the zero set of $\boldsymbol{f}$ provided that

$$\gamma(\boldsymbol{q}, \dot{\boldsymbol{q}}) = -\left(\frac{\dot{\boldsymbol{q}}\dot{\boldsymbol{q}}^\top}{\dot{\boldsymbol{q}}^\top M_{\mathcal{L}_e}\dot{\boldsymbol{q}}}\right)\partial\psi - \beta\dot{\boldsymbol{q}}. \tag{8}$$

Consequently, we aim to leverage this result by using $\boldsymbol{f}_\theta^{\mathcal{C}}$ as the system with the compatible potential. From the previous section, we already concluded that the zero set of this system maps to the equilibrium $\boldsymbol{x}_{\text{g}} \in \mathcal{T}$, which is a property we desire. Hence, it remains to find a compatible potential for this function to employ the result from Theorem III.5 [21].

Notably, if (4) is successfully minimized, it is possible to design a compatible potential for the system $\boldsymbol{f}_\theta^{\mathcal{T}}$ in the latent space $\mathcal{L}$ using the mapping $\boldsymbol{\rho}_\theta$ by setting $\dot{\boldsymbol{x}} = \boldsymbol{0}$. Specifically,

$$\psi(\boldsymbol{x}) = \|\boldsymbol{\rho}_\theta(\boldsymbol{x}_{\text{g}}, \boldsymbol{0}) - \boldsymbol{\rho}_\theta(\boldsymbol{x}, \boldsymbol{0})\|^2. \tag{9}$$

To observe that this is a compatible potential of $\boldsymbol{f}_\theta^{\mathcal{T}}$, first, we highlight that since $\boldsymbol{x}_{\text{g}}$ is asymptotically stable, we have $\partial\psi(\boldsymbol{x}_{\text{g}}) = \boldsymbol{0}$ if and only if $\boldsymbol{f}(\boldsymbol{x}_{\text{g}}, \boldsymbol{0}) = \boldsymbol{0}$. This satisfies the first condition of Definition 1. Second, we note that for all $\boldsymbol{x} \neq \boldsymbol{x}_{\text{g}}$, Eq. (4) enforces the value of $\psi(\boldsymbol{x})$ to decrease as $\boldsymbol{f}_\theta^{\mathcal{T}}$ evolves over time, provided that $\boldsymbol{\rho}_\theta$ has a Lipschitz constant that is not large, which can be controlled through regularization. Thus, this potential also satisfies the second condition of Definition 1. Finally, it only remains to express the gradient of this potential in configuration space, previously denoted as $\partial\psi$. For clarity, we will henceforth write this as $\partial\psi/\partial\boldsymbol{q}$. To achieve this, we require the forward kinematics from configuration space $\mathcal{C}$ to task space $\mathcal{T}$, denoted $\phi^{\mathcal{T}}$. Then, we obtain

$$\partial\psi = \frac{\partial\psi}{\partial\boldsymbol{q}} = \frac{\partial\psi}{\partial\boldsymbol{x}} \cdot J_{\phi^{\mathcal{T}}}(\boldsymbol{x}), \tag{10}$$

where $J_{\phi^{\mathcal{T}}}$ is the Jacobian matrix of the forward kinematics. The matrix $J_{\phi^{\mathcal{T}}}$ is commonly available in robotic frameworks, and the term $\partial\psi/\partial\boldsymbol{x}$ can be approximated via automatic differentiation tools for DNNs.

IV. Experimental Results

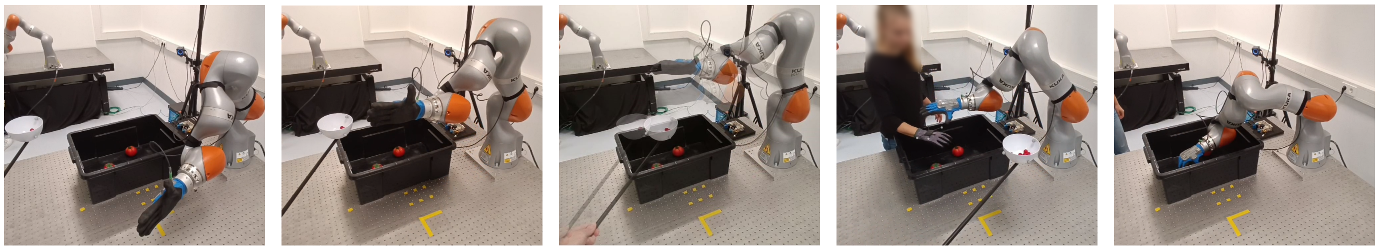
A. Experimental setup and performance metrics

To showcase the performance of the two variations of TamedPUMA, FPM and CPM, simulations using the Pybullet physics simulation [4] and real-world experiments are performed on a 7-DoF manipulator. A DNN is trained using 10 demonstrations where a tomato is picked within a crate. In Figure 1, an illustrative example of our proposed approaches compared against the baselines geometric fabrics and PUMA is provided for a point-mass.

The proposed FPM and CPM, are compared against vanilla geometric fabrics and vanilla PUMA. All are evaluated on their *success-rate* and *time-to-success* indicating the ratio of scenarios and required time respectively for the robot to reach the goal pose with a collision-free motion, given a margin of $\|\boldsymbol{x}_{\text{ee}} - \boldsymbol{x}_{\text{g}}\|_2 < 0.02$. The methodologies are also compared on the average *minimum clearance* between the collision shapes of the robot and obstacles over all scenarios and *computation time*. In addition, the *path difference* to the desired path by PUMA is denoted, $\|\boldsymbol{X}_{\text{ee}} - \boldsymbol{X}_{\text{PUMA}}\|_2$, where $\boldsymbol{X}$ indicates the stacked list of end-effector poses along the path.

TABLE I: Statistics for 10 simulated scenarios of the two proposed variations of TamedPUMA, namely FPM and CPM, compared against vanilla fabrics and vanilla PUMA. For completeness, the results of PUMA in an obstacle-free (PUMA$_{\text{free}}$) and obstacle-rich environment (PUMA$_{\text{obst}}$) are provided.

	Success-Rate	Time-to-Success [s]	Min Clearance [m]	Computation time [ms]	Path difference to PUMA
PUMA$_{\text{free}}$	1	3.77 ± 0.40	-	3.89 ± 0.26	0
PUMA$_{\text{obst}}$	0.2	3.81 ± 0.05	0.01 ± 0.03	4.77 ± 1.14	0
Fabrics	0.9	8.70 ± 5.43	0.05 ± 0.02	0.40 ± 0.061	0.18 ± 0.24
FPM	1	9.56 ± 7.62	0.04 ± 0.02	5.00 ± 0.74	0.13 ± 0.21
CPM	1	8.95 ± 4.13	0.05 ± 0.03	6.18 ± 0.74	0.11 ± 0.16



(a) Initial pose (b) Bowl approaches (c) Avoid the bowl (d) Avoid the hand (e) Goal reached

Fig. 2: Selected time frames of CPM during a tomato-picking task.

B. Simulation experiments on a 7-DOF manipulator

In simulation, 10 realistic scenarios are explored of the tomato-picking task varying the initial configuration and the location of the obstacles. For each scenario, at least one obstacle is situated between the initial configuration and goal pose. The start and goal pose of the end-effector are ensured to be collision-free and the goal pose is reachable with PUMA in an obstacle-free environment. The last five links on the robotic chain are considered for collision avoidance, given that the first three links have restricted movement due to the base being mounted to the table.

As depicted in Table I, the two variations of TamedPUMA improve the success rate with respect to PUMA$_{obst}$ as it allows for whole-body obstacle avoidance. In contrast to geometric fabrics, FPM and CPM can track the desired motion profile leading to a marginally smaller path difference with PUMA compared to geometric fabrics. In Fig. I, this path difference can be misleading as the path is desired to deviate from the path by PUMA if obstacles are avoided. However, if obstacles are not obstructing the path, FPM and CPM converge to the path provided by PUMA with a path difference of 0.04 ± 0.05 and 0.05 ± 0.05 respectively, while geometric fabrics is unable to follow the desired motion profile with a path difference to PUMA of 0.15 ± 0.19. Geometric fabrics also result in a deadlock in 1 of the 10 scenarios where the robot does not reach the goal as it is unable to move around the edge of the crate. In contrast, PUMA already encoded the intuitive trajectory of first moving the end-effector above the box, before performing a grasp. Although the CPM has stronger theoretical guarantees compared to FPM, performance is similar when comparing the two proposed approaches in Table I. Computation times are within the order of 4-7 ms making the methodologies well suitable for real-time reactive motion planning.

C. Real-world experiments on a 7-DOF manipulator

During experiments with the real 7-DOF manipulator, two dynamic obstacles blocked the way: a bowl with grapes and a person's hand. Both these obstacles are moving through the robot's workspace and are tracked in real-time via an optitrack system. In addition to the collision spheres considered during simulation, an additional collision sphere is added to the collision geometry on the center of the robotic hand. Snapshots of a real-world experiment of the CPM are illustrated in Fig. 2. If the obstacles are not blocking the trajectory of the robot, the observed behavior of the proposed methods, FPM and CPM, are similar to PUMA and showcases clearly the learned behavior as demonstrated by the human. The user can push the robot away from the goal and recover from this disturbance. In the presence of obstacles, FPM and CPM achieve collision avoidance between the considered links on the robot and the obstacles while reaching the goal pose as illustrated in Fig. 2.

A low-level joint impedance controller tracks the desired velocities and positions, outputting torque commands. This allows users to physically interact with the robot.

V. Conclusion

Imitation learning via stable motion primitives is a well-suitable approach for learning motion profiles from demonstrations while providing convergence to the goal. We introduced TamedPUMA, a safe and stable extension of learned stable motion primitives augmented with geometric fabrics for safe and stable operations in the presence of obstacles. We proposed two variations, the Forcing Policy Method and Compatible Potential Method, ensuring respectively that the goal is stable, or the stronger notion that the system converges towards the reachable goal. Experiments were carried out both in simulation and in the real world. When trained on a tomato-picking task, the proposed TamedPUMA generates a desired motion profile using a DNN while taking whole-body collision avoidance and joint limits into account.

Acknowledgments

This project has received funding from the European Union through ERC, INTERACT, under Grant 101041863, and by the NXTGEN national program. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the NXTGEN national program. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] David Bao, S-S Chern, and Zhongmin Shen. *An introduction to Riemann-Finsler geometry*, volume 200. Springer Science & Business Media, 2012.

[2] Aude Billard, Sina Mirrazavi, and Nadia Figueroa. *Learning for adaptive and reactive robot control: a dynamical systems approach*. Mit Press, 2022.

[3] Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, volume 49. Springer, 2019.

[4] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016.

[5] NB Figueroa Fernandez and A Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. *Proceedings of Machine Learning Research*, 2018.

[6] Yingbai Hu, Fares J Abu-Dakka, Fei Chen, Xiao Luo, Zheng Li, Alois Knoll, and Weiping Ding. Fusion dynamical systems with machine learning in imitation learning: A comprehensive overview. *Information Fusion*, page 102379, 2024.

[7] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[8] Noémie Jaquier, Leonel Rozo, Sylvain Calinon, and Mathias Bürger. Bayesian optimization meets riemannian manifolds in robot learning. In *Conference on Robot Learning*, pages 233–246. PMLR, 2020.

[9] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

[10] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5): 943–957, 2011.

[11] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[12] Holger Klein, Noémie Jaquier, Andre Meixner, and Tamim Asfour. A riemannian take on human motion analysis and retargeting. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5210–5217. IEEE, 2022.

[13] Holger Klein, Noémie Jaquier, Andre Meixner, and Tamim Asfour. On the design of region-avoiding metrics for collision-safe motion generation on riemannian manifolds. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2346–2353. IEEE, 2023.

[14] Andre Lemme, Klaus Neumann, René Felix Reinhart, and Jochen J Steil. Neural learning of vector fields for encoding stable dynamical systems. *Neurocomputing*, 141:3–14, 2014.

[15] Ge Li, Zeqi Jin, Michael Volpp, Fabian Otto, Rudolf Lioutikov, and Gerhard Neumann. Prodmp: A unified perspective on dynamic and probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 8(4): 2325–2332, 2023.

[16] Rodrigo Pérez-Dattari and Jens Kober. Stable motion primitives via imitation and contrastive learning. *IEEE Transactions on Robotics*, 39(5):3909–3928, 2023.

[17] Rodrigo Pérez-Dattari, Cosimo Della Santina, and Jens Kober. Deep metric imitation learning for stable motion primitives. *arXiv preprint*, 2023.

[18] Nicolas Perrin and Philipp Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59, 2016.

[19] Affan Pervez, Yuecheng Mao, and Dongheui Lee. Learning deep movement primitives using convolutional neural networks. In *2017 IEEE-RAS 17th international conference on humanoid robotics (Humanoids)*, pages 191–197. IEEE, 2017.

[20] Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In *Learning for Dynamics and Control*, pages 630–639. PMLR, 2020.

[21] Nathan Ratliff and Karl Van Wyk. Fabrics: A foundationally stable medium for encoding prior experience. *arXiv preprint:2309.07368*, 2023.

[22] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[23] Nathan D Ratliff, Karl Van Wyk, Mandy Xie, Anqi Li, and Muhammad Asif Rana. Optimization fabrics. *arXiv preprint arXiv:2008.02399*, 2020.

[24] Barry Ridge, Andrej Gams, Jun Morimoto, Aleš Ude, et al. Training of deep neural networks for the generation of dynamic movement primitives. *Neural Networks*, 127: 121–131, 2020.

[25] Max Spahn, Martijn Wisse, and Javier Alonso-Mora. Dynamic optimization fabrics for motion generation. *IEEE Transactions on Robotics*, 2023.

[26] Aleš Ude, Bojan Nemec, Tadej Petrić, and Jun Morimoto. Orientation in cartesian space dynamic movement primitives. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2997–3004. IEEE, 2014.

[27] J Urain, M Ginesi, D Tateo, and J Peters. Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5231–5237, 2020.

[28] Karl Van Wyk, Ankur Handa, Viktor Makoviychuk, Yujie Guo, Arthur Allshire, and Nathan D. Ratliff. Geometric fabrics: a safe guiding medium for policy learning. *arXiv preprint arXiv:2310.12831*, 2023.

[29] Mandy Xie, Karl Van Wyk, Anqi Li, Muhammad Asif Rana, Qian Wan, Dieter Fox, Byron Boots, and Nathan Ratliff. Geometric fabrics for the acceleration-based design of robotic motion. *arXiv preprint arXiv:2010.14750*, 2020.

[30] Mandy Xie, Anqi Li, Karl Van Wyk, Frank Dellaert, Byron Boots, and Nathan Ratliff. Imitation learning via simultaneous optimization of policies and auxiliary trajectories. *arXiv preprint arXiv:2105.03019*, 2021.

[31] Mandy Xie, Ankur Handa, Stephen Tyree, Dieter Fox, Harish Ravichandar, Nathan D Ratliff, and Karl Van Wyk. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In *Conference on Robot Learning*, pages 1355–1367. PMLR, 2023.