Incremental Skill Learning of Stable Dynamical Systems

Matteo Saveriano[1] and Dongheui Lee[1,2]

***Abstract*— Efficient skill acquisition, representation, and on-line adaptation to different scenarios has become of fundamental importance for assistive robotic applications. In the past decade, dynamical systems (DS) have arisen as a flexible and robust tool to represent learned skills and to generate motion trajectories. This work presents a novel approach to incrementally modify the dynamics of a generic autonomous DS when new demonstrations of a task are provided. A control input is learned from demonstrations to modify the trajectory of the system while preserving the stability properties of the reshaped DS. Learning is performed incrementally through Gaussian process regression, increasing the robot's knowledge of the skill every time a new demonstration is provided. The effectiveness of the proposed approach is demonstrated with experiments on a publicly available dataset of complex motions.**

I. Introduction

Future robots will have a tight interaction with humans and they will need an increased versatility to rapidly adapt their behaviour to dynamic and potentially unseen situations. Having a fixed set of predefined skills is not sufficient to execute everyday tasks in human populated environments. Programming by Demonstrations (PbD) is a well-established approach to rapidly teach new skills avoiding tedious programming [1], [2]. In the PbD framework, the robot can learn by observing the human behaviour (imitation learning) [3], [4], or an expert user can directly guide the robot towards the task execution (kinesthetic teaching) [5], [6].

Point-to-point motions, also called discrete movements, are spatial motions ending at a specified target. Discrete movements are of importance in several robotic applications, e.g. in assembly tasks, and they can be combined to build complex tasks [7]. Recent work in PbD [8]–[14] focuses on representing discrete movements as stable dynamical systems (DS), learned from human demonstrations. DS have been proven to be flexible enough to accurately represent complicated motions [11]–[13]. Moreover, robots driven by stable DS are guaranteed to reach the desired position, and can react in real-time to external perturbations, like changes in the target position or unexpected obstacles [15]–[19]. DS have been also used to learn impedance behaviors from demonstrations [20], [21] and to refine learned behaviors through reinforcement learning [22]–[24].

This work focuses on the incremental learning of point-to-point motions represented as stable dynamical system, i.e. on how to modify the robot's behavior as novel demonstrations of the task are provided. In our framework, we assume that a predefined skill is given in the form of a stable DS, the so-called original system. The trajectories of the original DS are modified by a reshaping control input, retrieved on-line by means of Gaussian process regression [25]. The reshaping action is learned incrementally from user demonstrations by adding new points to the training set. To alleviate the problem of the increasing computational time, a trajectory-based sparsity criteria is introduced to reduce the amount of novel points added to the training data. The reshaping controller guarantees an accurate reproduction of the demonstrated task without affecting the convergence properties of the original DS. Our formulation does not require any prior knowledge on the original DS and it applies to a wide class of non-linear, autonomous systems. An overview of the proposed Reshaped Dynamical Systems (RDS) is shown in Fig. 1.

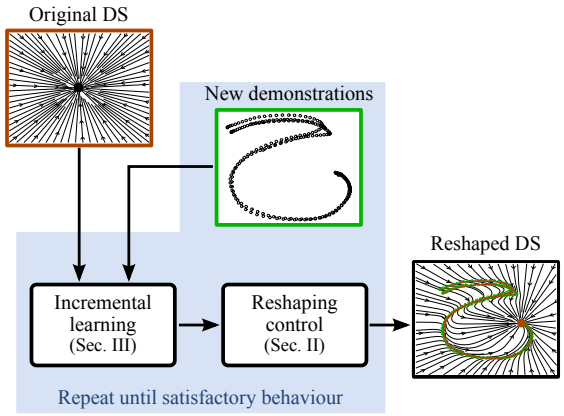


Fig. 1. Overview of the Reshaped Dynamical Systems (RDS) approach.

The rest of the paper is organized as follows. Section II presents an approach to modify the trajectory of a DS without affecting its stability. The proposed incremental learning algorithm is described in Section III. Section IV describes the related works. RDS is evaluated on a public dataset and compared with the state-of-the-art approaches [12], [26] in Section V. Section VI concludes the paper.

II. Dynamical System Reshaping

A. Problem definition

Assume that a robotic skill is encoded in a first-order DS

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}), \tag{1}$$

where[1] $\boldsymbol{x}, \dot{\boldsymbol{x}} \in \mathbb{R}^n$ are respectively the position and the velocity of the robot (in joint or Cartesian space), and $\boldsymbol{f} : \mathbb{R}^n \rightarrow$

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Weßling, Germany matteo.saveriano@dlr.de.

[2]Human-Centered Assistive Robotics, Technical University of Munich, Munich, Germany dhlee@tum.de.

This work has been supported by Helmholtz Association.

[1]We omit the time dependency of $\boldsymbol{x}$ to simplify the notation.

$\mathbb{R}^n$ is a continuous and continuously differentiable non-linear function. A solution of (1), namely $\boldsymbol{\Phi}(\boldsymbol{x}_0, t) \in \mathbb{R}^n$, is called trajectory. Different initial conditions $\boldsymbol{x}_0$ generate different trajectories. A point $\hat{\boldsymbol{x}} : \boldsymbol{f}(\hat{\boldsymbol{x}}) = \boldsymbol{0} \in \mathbb{R}^n$ is an equilibrium point. An equilibrium $\hat{\boldsymbol{x}} \in S \subset \mathbb{R}^n$ is locally asymptotically stable (LAS) if $\lim_{t\to+\infty} \boldsymbol{\Phi}(\boldsymbol{x}_0, t) = \hat{\boldsymbol{x}}, \forall \boldsymbol{x}_0 \in S$. If $S = \mathbb{R}^n$, $\hat{\boldsymbol{x}}$ is globally asymptotically stable (GAS) and it is the only equilibrium of the DS.

Given a novel demonstration of the task, our goal is to learn a control input that satisfies the following requirements:

1) The reshaped DS has the same GAS equilibrium of $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$.
2) The trajectories of the reshaped DS follow the given demonstrations $\mathcal{D}$.
3) The control input is incrementally updated as novel demonstrations are given.

A control structure that satisfies requirement 1) is presented in Sec. II-B. An approach to learn a control input that satisfies 2) and 3) is presented in Sec. III.

B. Globally stable reshaping controller

In order to modify the trajectories of (1), one can exploit a suitable control input $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^n$. Instead of considering the general form of controlled DS $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, we assume an additive and smooth (continuous and continuously differentiable) control input, obtaining the controlled DS $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x})$. In general, the controlled system is not guaranteed to converge to the same equilibrium point $\hat{\boldsymbol{x}}$ of (1). Hence, we exploit a clock signal to suppress the control input $\boldsymbol{u}(\boldsymbol{x})$ after t_f seconds, ensuring global convergence to $\hat{\boldsymbol{x}}$. The resulting reshaped DS can be written as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + s\boldsymbol{u}(\boldsymbol{x}) \tag{2a}$$

$$\dot{s} = \alpha(\hat{s} - s) \tag{2b}$$

where $\hat{s} = 1$ for $t \leq t_f$, $\hat{s} = 0$ for $t > t_f$, and the control input $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^n$ is a continuous and continuously differentiable function. The time $t_f > 0$, after which $\hat{s} = 0$, is a tunable parameter. The scalar gain $\alpha > 0$ determines how fast s reaches the desired value $\hat{s}$ and it can be tuned considering that, in practice, $s = \hat{s}$ after $5/\alpha$ seconds. In all the experiments we choose $\alpha = 10$ to have $s = \hat{s}$ after $0.5\,\mathrm{s}$.

The reshaped dynamics (2a)–(2b) is GAS if the original DS (1) is GAS, as stated by the following theorem.

Theorem 1: Assume that the dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$ in (1) has a GAS equilibrium $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ and that $\hat{s}$ in (2b) is $\hat{s} = 0$ for $t > t_f$. Under these assumptions the reshaped dynamics (2a) globally asymptotically converges to $\hat{\boldsymbol{x}}$.

Proof: Note that the linear dynamics of $\dot{s}$ in (2b) does not depend on the dynamics of $\dot{\boldsymbol{x}}$ in (2a). Being $\alpha > 0$ and $\hat{s} = 0$ for $t > t_f$, we can conclude that s converges to $\hat{s} = 0$ for $t \to +\infty$. Hence, for $t \to +\infty$, the term $s\boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{0}$ and $\boldsymbol{x}$ converges to $\hat{\boldsymbol{x}}$, the GAS equilibrium of (1). ■

The formulation introduced in (2a)–(2b) ensures that the robot's motion is generated by a stable (first-order) dynamical system. As discussed in Sec. I, stable DS generate converging motions that accurately reproduce the demonstrations and are robust to external perturbations. An additive control input is assumed in (2a). While this is a common assumption for many physical systems like robots, it also eases the computation of the training data as detailed in Sec. III-A. The clock signal in (2b) introduces a time dependency in the reshaped system. This makes easy to ensure stability properties (see Theorem 1) without losing some benefits of autonomous DS in case of external perturbations. Indeed, if the robot is blocked and time passes, the control input remains unchanged because it only depends on the robot position. When the robot is released, it smoothly continues its motion towards the goal.

The proposed reshaped DS (2a) resembles the dynamic movement primitives (DMPs) framework [8]. DMPs reshape a second-order linear system (original DS) with a non-linear forcing term (control input). An exponentially decaying clock signal is used to cancel the effects of the forcing term guaranteeing global stability. Compared to the original DMPs, our approach differs in the following aspects. In our framework the original DS can be any non-linear system. As experimentally shown in Sec. V, the adoption of a non-linear DS significantly improves the accuracy in reproducing complex, intrinsically non-linear movements. Moreover, the non-linear control input can be learned and incrementally updated from multiple demonstrations, while batch learning from a single demonstration is used in original DMPs. The adoption of multiple demonstrations improves the generalization capabilities of the learning algorithm.

III. Learning the reshaping controller

In this section, an approach is presented to learn and online retrieve the control input $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^n$ in (2a) for each state $\boldsymbol{x}$. We firstly describe how to compute the training data from the given demonstrations and learn the reshaping controller. Then, an approach is presented to incrementally update the reshaping controller.

A. Computation of the training data

Consider that a new demonstration of a skill is given as $\mathcal{D} = \{\boldsymbol{x}_d^t,\ \dot{\boldsymbol{x}}_d^t\}_{t=1}^{T_d}$, where $\boldsymbol{x}_d^t \in \mathbb{R}^n$ is the desired state vector (e.g. the robot position) at time t, $\dot{\boldsymbol{x}}_d^t \in \mathbb{R}^n$ is the time derivative of $\boldsymbol{x}_d^t$ (e.g. the robot velocity), and T_d is the number of samples. To learn the control input $\boldsymbol{u}(\boldsymbol{x})$ in (2a) from $\mathcal{D}$, demonstrations are first converted into a set of input/output training data.

Assuming $s = 1$ and considering (2a), the dynamics of $\dot{\boldsymbol{x}}$ in (2a) can be re-written as

$$\dot{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{u}(\boldsymbol{x}), \tag{3}$$

which shows that the desired control input $\boldsymbol{u}(\boldsymbol{x})$ is a non-linear mapping between $\boldsymbol{x}$ and $\dot{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{x})$. Hence, we consider the demonstrated states $\boldsymbol{X} = \{\boldsymbol{x}_d^t\}_{t=1}^{T_d}$ as input and $\boldsymbol{\Lambda} = \left\{\dot{\boldsymbol{x}}_d^t - \boldsymbol{f}(\boldsymbol{x}_d^t)\right\}_{t=1}^{T_d}$ as observations (output) of $\boldsymbol{u}(\boldsymbol{x})$. In other words, the learned controller adds a displacement to $\boldsymbol{f}(\boldsymbol{x})$ which makes the reshaped dynamics close to the demonstrated one (see requirement 2) in Sec. II-A).

Once the training data are computed, any regression technique can be applied to learn the reshaping controller and retrieve a smooth control input for each value of $\boldsymbol{x}$. In this work, we adopt a local regression technique, namely the Gaussian process (GP) regression [25]. Local regression ensures that $\boldsymbol{u} \to \mathbf{0}$ when the state is far from the demonstrated trajectories, making possible to locally follow the demonstrations leaving the rest of the trajectory almost unchanged. Note that GP does not require the alignment of input sequences to a common length, being the regression performed considering all the points in the training set.

B. Gaussian process regression

Gaussian processes (GP) [25] assume that the training input $\boldsymbol{X}$ and output $\boldsymbol{\Lambda}_i = \{\lambda_i^t\}_{t=1}^T$, where λ_i is the i-th component of $\boldsymbol{\Lambda}$, are drawn from the scalar noisy process $\lambda_i^t = g(\boldsymbol{x}^t) + \epsilon \in \mathbb{R},\ t = 1 \dots T_d$. The noise term ϵ is Gaussian with zero mean and variance σ_n^2, while $g(\cdot)$ is a smooth and unknown function. The joint distribution between training points and the output λ_i^* at a query point $\boldsymbol{x}^*$ is

$$\begin{bmatrix} \boldsymbol{\Lambda}_i \\ \lambda_i^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K}_{XX} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}_{x^*X} \\ \boldsymbol{K}_{Xx^*} & k(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix}\right), \quad (4)$$

where $\boldsymbol{K}_{x^*X} = \{k(\boldsymbol{x}^*, \boldsymbol{x}_1^t)\}_{t=1}^{T_d}$, $\boldsymbol{K}_{Xx^*} = \boldsymbol{K}_{x^*X}^{\mathrm{T}}$, $k(\cdot,\cdot)$ is a covariance function, and the element ij of the matrix $\boldsymbol{K}_{XX}$ is given by $\{\boldsymbol{K}_{XX}\}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

To make predictions, one can consider that the conditional distribution of λ_i^* given $\boldsymbol{\Lambda}_i$ can be written as

$$\begin{aligned} &\lambda_i^* | \boldsymbol{\Lambda}_i \sim \mathcal{N}\left(\mu_{\lambda_i^*|\boldsymbol{\Lambda}_i}, \sigma^2_{\lambda_i^*|\boldsymbol{\Lambda}_i}\right), \\ &\mu_{\lambda_i^*|\boldsymbol{\Lambda}_i} = \boldsymbol{K}_{x^*X}\left(\boldsymbol{K}_{XX} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Lambda}_i, \\ &\sigma^2_{\lambda_i^*|\boldsymbol{\Lambda}_i} = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{K}_{x^*X}\left(\boldsymbol{K}_{XX} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{K}_{Xx^*}. \end{aligned} \quad (5)$$

The mean $\mu_{\lambda_i^*|\boldsymbol{\Lambda}_i}$ is used as an estimate of λ_i^* given $\boldsymbol{\Lambda}_i$. Note that the described procedure holds for a scalar output. Hence, n GPs are used to represent the control input $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}^n$.

In our approach, $k(\cdot,\cdot)$ is the squared exponential function

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_k^2 \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2l}\right) + \sigma_n^2 \delta(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad (6)$$

where $\delta(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1$ if $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| = 0$ and $\delta(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0$ otherwise. The variance σ_k^2, the length scale l, and the noise sensitivity σ_n^2 are positive hyper-parameters which can be hand-tuned or learned from demonstrations [25]. The kernel function (6) guarantees that the control input goes to zero ($\boldsymbol{u}(\boldsymbol{x}) \to \mathbf{0}$) for points far from the demonstrated positions.

C. Incremental gaussian process updating

GP regression is computed considering all the training data. If, as in this work, GP hyper-parameters are fixed, incremental GP learning is performed by simply adding new points to the training set. To reduce the computational effort due to the matrix inversion in (5), incremental GP algorithms introduce criteria to sparsely represent incoming data [26], [27]. Following this idea and assuming that T_d data $\{\boldsymbol{x}_d^t, \dot{\boldsymbol{x}}_d^t - \boldsymbol{f}(\boldsymbol{x}_d^t)\}_{t=1}^{T_d}$ are already in the training set, we propose to add a new data point $[\boldsymbol{x}_d^{T_d+1}, \dot{\boldsymbol{x}}_d^{T_d+1} - \boldsymbol{f}(\boldsymbol{x}_d^{T_d+1})]$ if the cost is defined as

$$C^{T_d+1} = \|\dot{\boldsymbol{x}}_d^{T_d+1} - \boldsymbol{f}(\boldsymbol{x}_d^{T_d+1}) - \hat{\boldsymbol{u}}(\boldsymbol{x}_d^{T_d+1})\| > \bar{c}, \quad (7)$$

where $\hat{\boldsymbol{u}}$ is the control input predicted at $\boldsymbol{x}_d^{T_d+1}$ using only data $\{\boldsymbol{x}_d^t, \dot{\boldsymbol{x}}_d^t - \boldsymbol{f}(\boldsymbol{x}_d^t)\}_{t=1}^{T_d}$ already in the training set. The tunable parameter $\bar{c}$ represents the error in approximating the demonstrated state derivative and it can be easily tuned. For example, if $\dot{\boldsymbol{x}}_d$ is the robot velocity, $\bar{c} = 0.2$ means that velocity errors smaller than 0.2 m/s are acceptable.

IV. Related Work

A. Skills representation using dynamical systems

The dynamic movement primitive (DMP) framework [8] is one of the first examples of robotic skills representation via DS. DMPs exploit a non-linear forcing term, learned from a single demonstration, to reshape a linear dynamics, and a clock signal to suppress the non-linear force after a certain time guaranteeing the convergence towards the target. Task-parameterized motion primitives [9], [28] extend the standard DMP by introducing extra task-dependent parameters useful to adapt robot movements to novel scenarios.

The stable estimator of dynamical systems (SEDS) in [10] generates stable motions from a non-linear DS, represented by GMM. Global stability is ensured by constraining the GMM parameters to satisfy a set of stability constraints derived from a quadratic Lyapunov function. The main advantage of SEDS is that the learned system is globally stable. The main limitation is that contradictions may occur between the demonstrations and the quadratic stability constraints, preventing an accurate learning of the desired motion.

The accuracy problem is explicitly considered in several works [11]–[14], showing that complex motions can be accurately represented by non-linear DS. In [11], [12] two different approaches are proposed to learn a Lyapunov function which minimizes the contradictions between the stability constraints and the training data, favoring an accurate reproduction of complex motions. [13] learns a diffeomorphic transformation that projects the training data into a space where they are well represented by a quadratic Lyapunov function. Perrin et al. [14] propose a fast algorithm to learn diffeomorphic transformations from a single demonstration.

RDS is complementary to the state-of-the-art approaches for skill representation via DS. RDS, in fact, incrementally reshapes the trajectories of a given DS (original DS) without affecting its stability properties. The original DS can be either designed by an expert or learned from demonstrations using one of the the aforementioned approaches.

B. Incremental learning of robotic skills

Several approaches have been proposed to extend the DMP framework to incremental learning scenarios [29]–[34]. In [29] the recursive least square and a forgetting factor are used to incrementally update the DMP weights. Gams et al. [30] present a two-layered system for incremental learning of periodic movements. The first layer of the system is a DS which extracts the fundamental frequency of the

demonstrations. The second layer is a periodic DMP which learns the waveform of the demonstrated motion. The overall system works on-line, but it is limited to periodic motions, while discrete movements are the focus of our work. The work in [31] considers incremental human coaching for DMP. In the teaching phase, the user is considered as an obstacle, avoided by adding an extra forcing term to the DMP [19]. In this way, the human is able to modify on-line the robot's path without touching it. The novel path is used to incrementally updated DMP weights via recursive least square. Nemec et al. [32] leverage iterative learning control [35] to realize a learning strategy which is faster and more robust than recursive least square. The approaches in [31], [32] are evaluated on periodic movements, but they are also applicable to point-to-point motions. The interaction between two agents, modeled via DMPs, is incrementally learned in [33] to guarantee that both agents equilibrate into a common target, i.e. the two agents are effectively helping each other. Maeda et al. [34] propose active incremental learning with DMP and Gaussian Processes (GP). They learn a GP from demonstrations and use GP regression to retrieve a confidence execution bound. If the confidence bound is low, the robot explicitly asks for novel demonstrations and updates the GP weights. A DMP is then trained over the GP mean to generate a converging trajectory.

Similarly to DMP, RDS exploits an additive control input and a clock signal to reshape an asymptotically stable DS. The role of the clock signal in RDS and DMP is the same: suppress the control input to guarantee asymptotic stability. In DMP, the control input (or forcing term) is a function of time, while in RDS it is a function of the robot's position. A position dependent control generates smooth motions in case the robot is kept fixed by an external perturbation (see Sec. II-B). In the same situation, a time depended forcing term may generate big accelerations when the robot is released due to the time passed. DMP reshapes only linear spring-damper DS, while the proposed RDS applies to any autonomous DS. This is the main limitation of DMP-based incremental approaches. Indeed, linear DS generate straight trajectories and, as experimentally demonstrated in Sec. V-A, transforming a straight line into a non-linear path is not trivial and may generate a loss of accuracy, i.e. the learned motion does not accurately match the demonstrated one.

The work in [36] leverages Contraction theory to automatically compute a stabilizing control input for a DS represented by GMM. Even if the control input can be computed on-line, [36] only works for DS represented by GMM, while RDS applies to any parameterization. The Locally Modulated Dynamical Systems (LMDS) in [26] reshapes an autonomous DS using a modulation matrix, obtained by multiplying a rotation matrix by a scalar gain. The modulation matrix is incrementally learned from demonstrations using GP [25]. The learned modulation matrix does not generate any spurious attractor in the modulated DS. Moreover, the effects of the modulation disappear for points far from the demonstrations. These properties of the modulation matrix guarantee the local stability of the modulated DS. Even if global stability is not proved, experiments show that the modulated DS remains stable in practice. LMDS shares some similarities with the proposed RDS. Like RDS, LMDS applies to any autonomous DS (both linear and non-linear), it allows incremental learning from multiple demonstrations, and it permits an accurate reproduction of demonstrated trajectories. These similar features make interesting to experimentally compare the performance of RDS and LMDS (see Sec. V).

V. Results and comparisons

Experiments in this section show the effectiveness of the proposed Reshaped Dynamical Systems (RDS) approach. The LASA Handwriting dataset[2] is used as a benchmark. The dataset consists of 26 different point-to-point 2D motions, where each motion trajectory is demonstrated seven times and contains $T_d = 1000$ positions and velocities. All the demonstrations end at the target position $\hat{\boldsymbol{x}} = [0, 0]^{\mathrm{T}}$.

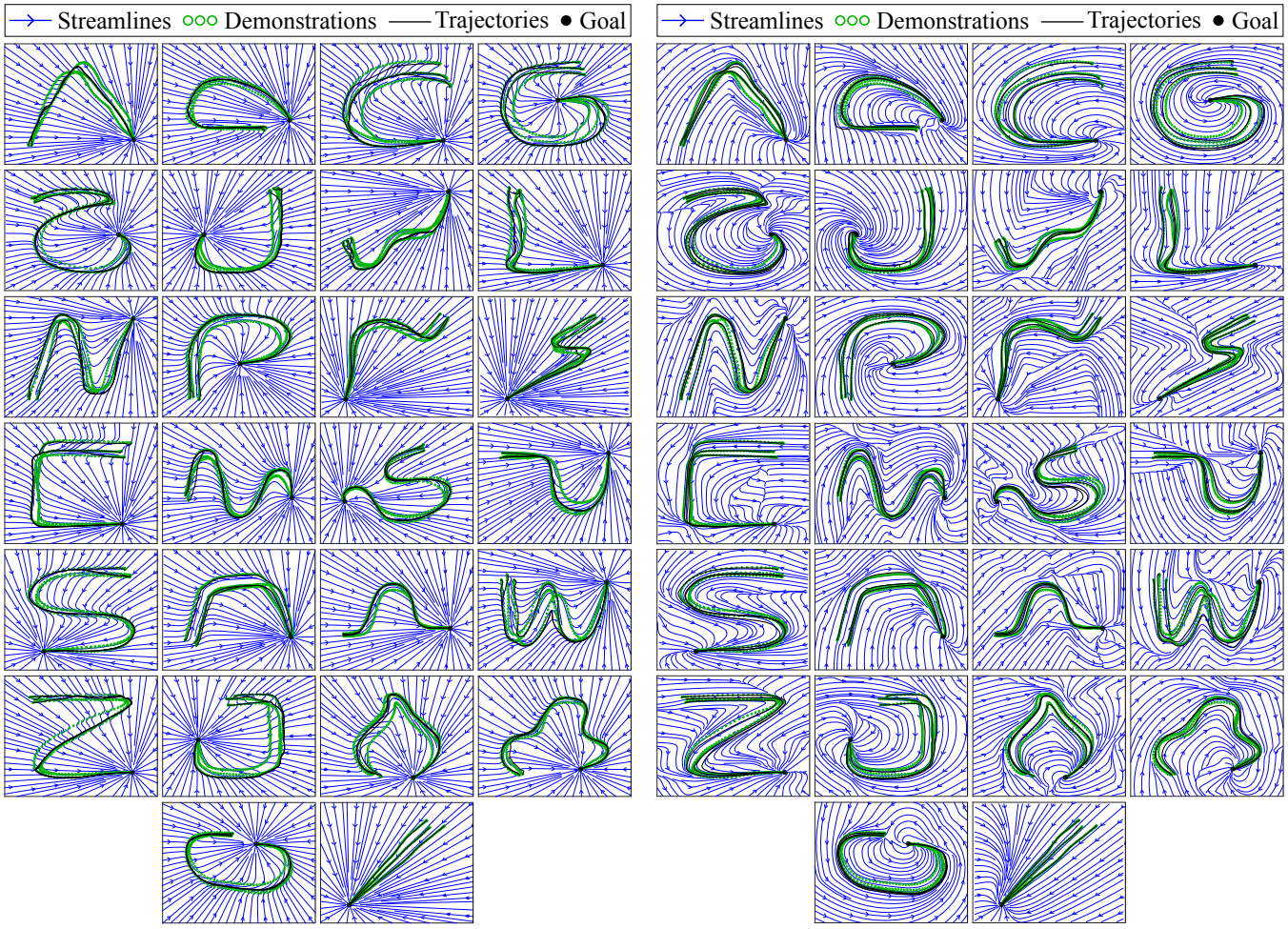
A. Accuracy test

This experiment compares the reproduction accuracy of the proposed RDS approach and the LMDS approach in [26]. As a proof of concepts, we consider the first three demonstrations for each motion in the LASA dataset and we subsample each demonstrated trajectory to 100 samples. LMDS guarantees local stability if the modulation is not active in a neighborhood of the equilibrium. This property is called locality in [26]. In order to ensure the locality property, we remove the last 10 points in each demonstration, creating a neighborhood of the origin without training points. We do the same with our approach for a fair comparison. To guarantee the maximum accuracy for both the approaches, all the 90 samples of each trajectory are considered without applying the sparsity criteria in Sec. III-C. Moreover, the optimal hyper-parameters are learned from the given demonstrations by maximizing the marginal log-likelihood [25].

The error that occurs when reproducing a demonstrated motion is measured by the swept error area (SEA) [12], defined as $SEA = \sum_{t=1}^{T_d-1} \mathcal{A}(\boldsymbol{x}_e^t, \boldsymbol{x}_e^{t+1}, \boldsymbol{x}_d^t, \boldsymbol{x}_d^{t+1})$, where $\boldsymbol{x}_e^t$ and $\boldsymbol{x}_d^t$ are respectively the reproduced and the demonstrated position at time t, T_d is the length of the demonstration, and $\mathcal{A}(\cdot)$ is the area of the tetragon formed by $\boldsymbol{x}_e^t$, $\boldsymbol{x}_e^{t+1}$, $\boldsymbol{x}_d^t$, and $\boldsymbol{x}_d^{t+1}$. The reproduced trajectory is equidistantly re-sampled to contain exactly T_d points. The SEA metric measures how well the DS preserves the shape of the demonstrations. To measure how the DS preserves the kinematics of the demonstrations, we use the velocity error defined in [11] as $V_{rmse} = \sqrt{\frac{1}{T_d}\sum_{t=1}^{T_d} \|\dot{\boldsymbol{x}}_d^t - \boldsymbol{f}(\boldsymbol{x}_d^t)\|^2}$, evaluated on the training data $\{\boldsymbol{x}_d^t, \dot{\boldsymbol{x}}_d^t\}_{t=1}^{T_d}$. The V_{rmse} measures the difference between the demonstrated velocities and the velocities generated by the learned DS for each training position.

Two different scenarios are considered. In the first scenario, the first-order, linear DS $\dot{\boldsymbol{x}} = -3\boldsymbol{x}$ is used as an original system. This task is particularly challenging, since a linear dynamics has to be transformed into the complex, non-linear motions of the LASA dataset (see the qualitative

[2]Available on-line: http://bitbucket.org/khansari/lasahandwritingdataset.



(a) Linear DS + RDS

(b) SEDS + RDS

Fig. 2. The linear DS $\dot{\boldsymbol{x}} = -3\boldsymbol{x}$ (a) and the SEDS non-linear system (b) reshaped into the 26 complex motions of the LASA dataset. The proposed RDS is able to learn complex motions without affecting the global stability of the original DS.

results in Fig. 2(a)). In the second scenario, a stable non-linear DS for each motion is learned by means of the Stable Estimator of Dynamical Systems (SEDS) approach in [10]. RDS and LMDS are applied to the learned DS to improve the reproduction accuracy of the SEDS algorithm.

The reproduction errors for the considered scenarios are shown in Tab. I. Since the reproduction errors (SEA and V_{rmse}) of each motion are not normally distributed, we consider the median M_e instead of the mean. To indicate the maximal and minimal deviation from the typical performance, we provide the location of the 10% (Q_{10}) and the 90% (Q_{90}) quantiles. As shown in Tab. I, SEDS+RDS has median errors of $81\,\text{mm}^2$ (SEA) and $2\,\text{mm/s}$ (V_{rmse}), which is significantly more accurate than Linear+RDS ($124\,\text{mm}^2$ for SEA and $5.2\,\text{mm/s}$ for V_{rmse}). Also with LMDS, modulating a SEDS system gives more accurate results than modulating a linear DS. This is an expected result, because it is easier to transform a dynamics which is close to the demonstrated motion, rather than transforming a linear DS into a complex motion. In both cases, RDS exhibits higher accuracy than LMDS, meaning that RDS is more effective in imposing a (potentially) different dynamics to a given DS.

TABLE I

REPRODUCTION ERROR OF RDS AND LMDS ON THE LASA DATASET.

Learning Method	SEA [mm^2] (M_e / Q_{10} / Q_{90})	V_{rmse} [mm/s] (M_e / Q_{10} / Q_{90})
Linear + RDS	**124 / 25 / 604**	**5.2** / 3.4 / **9.0**
Linear + LMDS	233 / 44 / 842	6.6 / **3.2** / 32.4
SEDS	366 / 73 / 584	11.5 / 8.2 / 30.9
SEDS + RDS	**81 / 27 / 251**	**2.0 / 1.0 / 3.2**
SEDS + LMDS	195 / 66 / 437	6.2 / 2.4 / 13.3

B. Incremental learning of multi-model behaviors

The goal of this experiment is two-fold. First, it shows that RDS can learn multi-model motions, i.e. different behaviours in different regions of the space. Second, the experiment shows that RDS and the batch learning approaches in Sec. IV-A are complementary. As a proof of concepts, we investigate the combination of RDS with SEDSII [12].

SEDSII computes a stabilizing control input from a

learned control Lyapunov function (CLF). The CLF is parameterized as $CLF = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}^0\boldsymbol{x} + \sum_{l=1}^{k}\beta^k(\boldsymbol{x})(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}^k(\boldsymbol{x}-\boldsymbol{\mu}^k))^2$, where β^k, $\boldsymbol{P}^k$, and $\boldsymbol{\mu}^k$ are learned from demonstrations by solving a constrained optimization problem. SEDSII is very effective in accurately learning complex motions while guaranteeing the convergence towards a unique target, but it is not prone to an incremental implementation. Hence, SEDSII is combined with RDS as

$$\dot{\boldsymbol{x}} = \underbrace{\boldsymbol{f}_{orig}(\boldsymbol{x})}_{DS} + \underbrace{\boldsymbol{u}_{CLF}(\boldsymbol{x})}_{SEDSII} + \underbrace{s\boldsymbol{u}_{RDS}(\boldsymbol{x})}_{RDS}, \tag{8}$$

where $\boldsymbol{f}_{orig}(\boldsymbol{x})$ is a possibly unstable DS, $\boldsymbol{u}_{CLF}(\boldsymbol{x})$ is the control input that stabilizes the original DS, and $s\boldsymbol{u}_{RDS}(\boldsymbol{x})$ is the reshaping controller defined in (2a)–(2b).

The controlled DS in (8) is tested in an incremental learning scenario to show the benefits of combining SEDSII and RDS. We consider the four multi-model motions of the LASA dataset. Each multi-model motion contains demonstrations of two or three different motions (see Fig. 3). Only the first demonstration of each different motion is considered in this experiment. Demonstrations are sub-sampled to 100 samples. The original DS $\boldsymbol{f}_{orig}(\boldsymbol{x})$ is a Gaussian process, learned from the given demonstration. The parameters used in this experiment are listed in Tab. II.

Three different tests are performed, as shown in Fig. 3. In the first case both $\boldsymbol{f}_{orig}(\boldsymbol{x})$ and $\boldsymbol{u}_{CLF}(\boldsymbol{x})$ are learned from the first motion (green circles in Fig. 3), while $\boldsymbol{u}_{RDS}(\boldsymbol{x}) = \boldsymbol{0}$. Novel demonstrations are then provided (brown and red circles) and $\boldsymbol{f}_{orig}(\boldsymbol{x})$ is incrementally updated as proposed in Sec. III-C. The CLF is not re-trained, since CLF parameters estimation cannot be performed efficiently. As shown in Fig. 3(a) and Tab. III, novel demonstrations are poorly represented, especially if different motions are demonstrated with the initial CLF parameters. In the second case, instead, the CLF is re-trained considering all the demonstrations. SEDSII accurately learns the multi-model motions, but the training takes almost 200 times longer. The third case shows the combination of SEDSII and RDS. Instead of re-training the CLF parameters, which is computationally expensive, the reshaping term $\boldsymbol{u}_{RDS}(\boldsymbol{x})$ is incrementally learned. With the parameters in Tab. II, the incremental learning approach in Sec. III-C uses from 45% to 65% of the points to encode the motion. Results in Fig. 3(c) and Tab. III clearly show that the combination of SEDSII and RDS is a good compromise in terms of accuracy and training time.

C. Incremental learning in higher dimensions

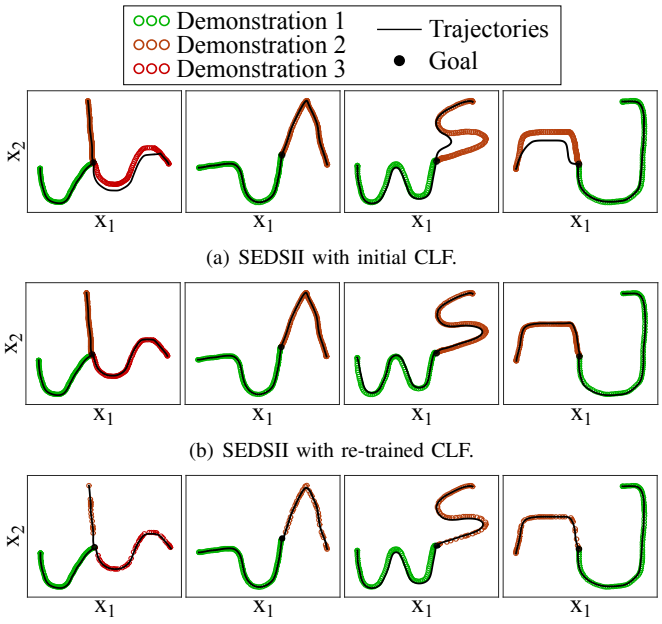
RDS is directly applicable to spaces of any dimension. On the contrary, LMDS exploits a rotation matrix, and defining a rotation in spaces with more than 3 dimensions is still an open problem (see Sec. V-D). Extending LMDS to high dimensional spaces is beyond the scope of this paper. Hence, in this experiment, we show the scalability of RDS to high dimensional spaces. To this end, we exploit the DS $\dot{\boldsymbol{x}} = 3(\hat{\boldsymbol{x}} - \boldsymbol{x})$ to generate a converging trajectory in a 6 dimensional space. The 6D state vector $\boldsymbol{x} = [\theta_1, \ldots, \theta_6]^{\mathrm{T}} \in \mathbb{R}^6$ can be interpreted as the joint angles of a robotic manipulator. The original DS generates a point-to-point motion in the joint space from $\boldsymbol{x}(0) = [35, 55, 15, -65, -15, 50]^{\mathrm{T}}$ deg to $\hat{\boldsymbol{x}} = [-60, 30, 30, -70, 25, 85]^{\mathrm{T}}$ deg. The original joint angles trajectories are shown in Fig. 4 (black solid lines).

The original trajectory is modified by providing 100 additional data in the time interval $[0.25, 1.25]$ s (brown solid line in Fig. 4). For each joint angle θ_i, the training data belongs to a straight line starting from $\theta_i(0.25)$ deg and ending at $\theta_i(0.25) + 20$ deg. As shown in Fig. 4, the reshaped trajectories (blue solid lines) accurately follow the given demonstration and converge to the desired goal $\hat{\boldsymbol{x}}$. Results are obtained with noise variance $\sigma_n^2 = 0.04$, signal variance $\sigma_k^2 = 0.1$, length scale $l = 0.01$, and the threshold $\bar{c} = 1$ rad/s. With the adopted $\bar{c}$ only 26 points over 100 are used to learn the control input.

TABLE II

PARAMETERS USED IN THE MULTI-MODEL BEHAVIORS LEARNING EXPERIMENT.

Original GP			SEDSII	RDS				
σ_k^2	σ_n^2	l	k (# CLF)	σ_k^2	σ_n^2	l	$\hat{c}$ [m/s]	t_f [s]
1.0	0.4	3.0	3	1.0	0.4	3.0	0.01	10.0



(a) SEDSII with initial CLF.

(b) SEDSII with re-trained CLF.

(c) SEDSII (initial CLF) combined with our reshaping approach.

Fig. 3. Qualitative results for the incremental learning of stable multi-model motions with different approaches.

TABLE III

REPRODUCTION ERRORS AND TRAINING TIMES OF RDS AND SEDSII ON THE MULTI-MODEL LASA MOTIONS.

Learning Method	Re-train CLF	SEA [mm^2] (M_e / Q_{10} / Q_{90})	Time [s] (mean ± std)
GPR + SEDSII	No	1.69 / 1.05 / 5.27	**0.009** ± **0.007**
GPR + SEDSII	Yes	**1.54** / 1.08 / **2.5**	3.6 ± 2.6
GPR + SEDSII + RDS	No	1.56 / **1.05** / 2.71	0.018 ± 0.013

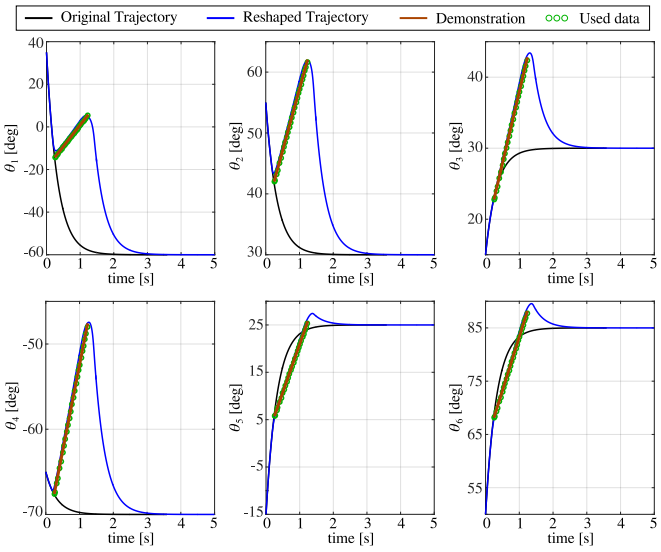


Fig. 4. Results obtained when RDS reshapes a motion in a 6D space.

D. Discussion

Performed experiments have underlined several properties of the proposed RDS approach. As shown in Sec. V-B, RDS can be easily combined with batch learning based approaches like SEDSII. The result of this combination is a system capable to incrementally refine a learned skill by significantly reducing training time while preserving the stability of the motion and the reproduction accuracy. Results in Sec. V-A show that reshaping a non-linear system results in more accurate reproduction than reshaping a linear DS. This is because it is hard to transform linear dynamics (straight lines) into a complex, intrinsically non-linear motion like the ones contained in the LASA dataset (see Fig. 2). It is worth noticing that DMPs also reshape a linear dynamical system and they may suffer from a similar accuracy problem.

RDS, as DMP, exploits a clock signal to suppress the control input after t_f seconds and to guarantee global stability. The value of t_f affects the obtained results. Small values of t_f may result in the loss of accuracy, if the control input is suppressed too early. On the contrary, too large values of t_f may cause the system to stop in a local equilibrium for long time before the control is deactivated. These problems were not encountered in our experiments. The reason is that we selected large values of t_f (larger than the demonstration time) and, since we used local demonstrations (i.e. no training data were added in a neighborhood of the equilibrium) and a local regression technique (GP), the control input was already vanishing ($\boldsymbol{u} \to \boldsymbol{0}$) for $t < t_f$. In other words, the reshaped DS was reaching the goal before t_f seconds.

In order to illustrate the behavior of RDS, we design a "failure" case where the reshaped system falls into a spurious attractor. Consider that RDS generates a spurious equilibrium if and only if, for $s = 1$, $\boldsymbol{u}(\boldsymbol{x}) = -\boldsymbol{f}(\boldsymbol{x})$ for some $\boldsymbol{x} \neq \hat{\boldsymbol{x}}$, i.e. if the original dynamics and the learned control are anti-parallel and have the same magnitude. To reproduce this situation, we reshape the 2D DS $\dot{\boldsymbol{x}} = -3\boldsymbol{x}$, used to generate a converging motion from $\boldsymbol{x}(0) = [2, 2]^{\mathrm{T}}$ m to $\hat{\boldsymbol{x}} = [0, 0]^{\mathrm{T}}$ m. As shown in Fig. 5, a demonstration is provided in the form of a straight line starting from the goal ($\hat{\boldsymbol{x}} = [0, 0]^{\mathrm{T}}$ m) and ending at $\boldsymbol{x} = [0.6, 0.6]^{\mathrm{T}}$ m, therefore pushing away the original DS from its global equilibrium. In this case, RDS generates a spurious attractor at about $\bar{\boldsymbol{x}} = [0.6, 0.6]^{\mathrm{T}}$ m (Fig. 5 (left)) because $\boldsymbol{u}(\bar{\boldsymbol{x}}) = -\boldsymbol{f}(\bar{\boldsymbol{x}})$. Satisfying the condition $\boldsymbol{u}(\boldsymbol{x}) = -\boldsymbol{f}(\boldsymbol{x})$ for $\boldsymbol{x} \neq \hat{\boldsymbol{x}}$ is improbable in realistic cases, as experimentally shown in this section. For instance, in Sec. V-C we also reshape $\dot{\boldsymbol{x}} = -3\boldsymbol{x}$ (in a 6D space) without generating spurious equilibria (Fig. 4). However, even if the motion temporary stops in a spurious equilibrium, the control input starts to vanish ($s \to 0$) for $t > t_f$ and the motion converges to the global attractor (Fig. 5 (right)). Depending on the application, waiting t_f seconds in a spurious attractor may be undesirable. Spurious attractors in first-order DS can be detected by checking if the DS velocity vanishes for any $\boldsymbol{x} \neq \hat{\boldsymbol{x}}$ and escaped by adding a small velocity in a fixed direction [15], e.g. a velocity pointing towards the goal. Moreover, the influence of t_f on the generated motion can be reduced by implementing a time scaling approach similar to the one exploited in DMPs [8].

RDS has been compared with LMDS, a prominent approach in the field. The comparison has shown that trajectories generated by RDS follow the demonstrations in a more accurate manner. The higher accuracy of RDS mainly depends on the fact that an additive control input is probably more effective in imposing a different dynamics to the original system. By inspecting (3), in fact, it is clear that the learned control $\boldsymbol{u}(\boldsymbol{x})$ cancels out the original system dynamics $\boldsymbol{f}(\boldsymbol{x})$ to impose the demonstrated dynamics $\dot{\boldsymbol{x}}_d$. As shown in Sec. V-C, RDS has the advantage to be directly applicable to high dimensional spaces, while LMDS requires the computation of a suitable rotation matrix. In spaces with more than three dimensions multiple parameterizations of a rotation are possible and all require at least $n(n-1)/2$ parameters [37], while the control input in RDS is a uniquely defined n-dimensional vector. As a final remark, recall that LDMS can encode periodic movements, while generating stable periodic orbits with RDS is still an open problem.

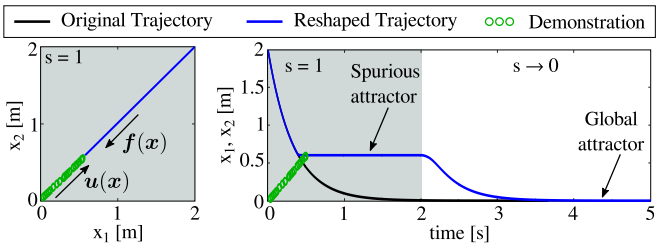


Fig. 5. Results obtained when the novel demonstration forces RDS to generate a spurious attractor. (Left) RDS generates a spurious attractor at $\bar{\boldsymbol{x}} = [0.6, 0.6]^{\mathrm{T}}$ because $\boldsymbol{u}(\bar{\boldsymbol{x}}) = -\boldsymbol{f}(\bar{\boldsymbol{x}})$. (Right) The reshaped trajectory stops into the spurious equilibrium until the control input is deactivated ($t > t_f = 2$ s) and then converges to the global equilibrium.

VI. Conclusions and Future Work

We presented the Reshaped Dynamical Systems (RDS), an approach useful to incrementally update a predefined skill by providing novel demonstrations. RDS is able to modify the trajectory of a dynamical system to follow demonstrated trajectories, while preserving eventual stability properties. To this end, a suitable control input is learned from demonstrations and retrieved on-line using Gaussian process regression. The procedure is incremental, meaning that the user can add novel demonstrations until the reproduced skill is satisfactory. Experimental results show the effectiveness of the proposed approach in reshaping dynamical systems. Compared to the state-of-the-art approaches, our method has a higher reproduction accuracy and it is directly applicable to high dimensional spaces.

RDS exploits Gaussian process regression to learn and retrieve the additive control input. Gaussian processes use all the training data to regress the output, which is a drawback in incremental learning scenarios where novel demonstrations are continuously provided. The problem is alleviated in this work by using a selection algorithm that limits the number of training data. However, the applicability of other incremental learning techniques to DS reshaping has not been investigated and it will be the topic of our future research.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds., 2008, pp. 1371–1394.

[2] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: a Reference*, P. Vadakkepat and A. Goswami, Eds. Springer, 2018, pp. 1–52.

[3] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 286–298, 2007.

[4] D. Lee and Y. Nakamura, "Mimesis model from partial observations for a humanoid robot," *The International Journal of Robotics Research*, vol. 29, no. 1, pp. 60–80, 2010.

[5] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2, pp. 115–131, 2011.

[6] M. Saveriano, S. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *International Conference on Robotics and Automation*, 2015, pp. 3570–3575.

[7] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human-robot interaction," *Autonomous Robots*, 2018.

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

[9] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

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

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

[12] S. M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[13] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robotics and Autonomous Systems*, vol. 70, pp. 1–15, 2015.

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

[15] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[16] M. Saveriano and D. Lee, "Point cloud based dynamical system modulation for reactive avoidance of convex and concave obstacles," in *International Conference on Intelligent Robots and Systems*, 2013, pp. 5380–5387.

[17] ——, "Distance based dynamical system modulation for reactive avoidance of moving obstacles," in *International Conference on Robotics and Automation*, 2014, pp. 5618–5623.

[18] M. Saveriano, F. Hirt, and D. Lee, "Human-aware motion reshaping using dynamical systems," *Pattern Recognition Letters*, vol. 99, pp. 96–104, 2017.

[19] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *International Conference on Robotics and Automation*, 2009, pp. 1534–1539.

[20] S. Calinon, I. Sardellitti, and D. Caldwell, "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies," in *International Conference on Intelligent Robots and Systems*, 2010, pp. 249–254.

[21] M. Saveriano and D. Lee, "Learning motion and impedance behaviors from human demonstrations," in *International Conference on Ubiquitous Robots and Ambient Intelligenc*, 2014, pp. 368–373.

[22] P. Kormushev, S. Calinon, and D. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *International Conference on Intelligent Robots and Systems*, 2010, pp. 3232–3237.

[23] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[24] F. Winter, M. Saveriano, and D. Lee, "The role of coupling terms in variable impedance policies learning," in *International Workshop on Human-Friendly Robotics*, 2015.

[25] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[26] K. Kronander, S. M. Khansari Zadeh, and A. Billard, "Incremental motion learning with locally modulated dynamical systems," *Robotics and Autonomous Systems*, vol. 70, pp. 52–62, 2015.

[27] L. Csató, "Gaussian processes - iterative sparse approximations," Ph.D. dissertation, Aston University, 2002.

[28] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of gmms," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 61–78, 2018.

[29] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[30] A. Gams, A. J. Ijspeert, S. Schaal, and J. Lenarčič, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Autonomous Robots*, vol. 27, no. 1, pp. 3–23, 2009.

[31] T. Petrič, A. Gams, L. Zlajpah, A. Ude, and J. Morimoto, "Online approach for altering robot behaviors based on human in the loop coaching gestures," in *International Conference on Robotics and Automation*, 2014, pp. 4770–4776.

[32] B. Nemec, T. Petrič, and A. Ude, "Force adaptation with recursive regression iterative learning controller," in *International Conference on Intelligent Robots and Systems*, 2015, pp. 2835–2841.

[33] T. Kulvicius, M. Biehl, M. J. Aein, M. Tamosiunaite, and F. Wörgötter, "Interaction learning for dynamic movement primitives used in cooperative robotic tasks," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1450–1459, 2013.

[34] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 37–46.

[35] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control," *Control Systems Magazine*, vol. 25, no. 3, pp. 96–114, 2006.

[36] C. Blocher, M. Saveriano, and D. Lee, "Learning stable dynamical systems using contraction theory," in *nternational Conference on Ubiquitous Robots and Ambient Intelligence*, 2017, pp. 124–129.

[37] D. Mortari, "On the rigid rotation concept in n-dimensional spaces," *Journal of the Astronautical Sciences*, vol. 49, no. 3, pp. 401–420, 2001.