Incremental Motion Reshaping of Autonomous Dynamical Systems

Matteo Saveriano[1]* and Dongheui Lee[2,3]

[1] University of Innsbruck, Innsbruck, Austria,
matteo.saveriano@uibk.ac.at
[2] Technical University of Munich, Munich, Germany
[3] German Aerospace Center (DLR), Weßling, Germany
* This work was carried out when the author was at the Human-centered Assistive Robotics, Technical University of Munich

Abstract. This paper presents an approach to incrementally learn a reshaping term that modifies the trajectories of an autonomous dynamical system without affecting its stability properties. The reshaping term is considered as an additive control input and it is incrementally learned from human demonstrations using Gaussian process regression. We propose a novel parametrization of this control input that preserves the time-independence and the stability of the reshaped system, as analytically proved in the performed Lyapunov stability analysis. The effectiveness of the proposed approach is demonstrated with simulations and experiments on a real robot.

Keywords: Incremental learning of stable motions · Dynamical systems for motion planning

1 Introduction

In unstructured environments, the successful execution of a task may depend on the capability of the robot to rapidly adapt its behavior to the changing scenario. Behavior adaptation can be driven by the robot's past experience or, as in the Programming by Demonstration (PbD) paradigm [20], by a human instructor that shows to the robot how to perform the task [16, 19, 21]. Demonstrated skills can be encoded in several ways. Dynamical systems (DS) are a promising approach to represent demonstrated skills and plan robotic motions in real-time. DS have been successfully used in a variety of robotic applications including point-to-point motion planning [3,6,14,17], reactive motion replanning [1,2,5,7], and learning impedance behaviors from demonstrations [18, 23].

Although DS are widely used in robotics applications, there is not much work on incremental learning approaches for DS. Some approaches have extended the Dynamic Movement Primitives (DMPs) [6] towards incremental learning. In [8], authors provide a corrective demonstration to modify a part of a DMP trajectory while keeping the rest unchanged. The work in [9, 10] propose passivity-based

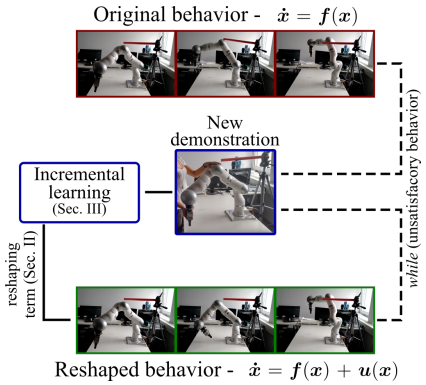


Fig. 1. System overview. (Top) The user observes the original robot behavior. (Middle) If the robot behavior does not match the requirements (it hits the red bar in the depicted case), novel demonstrations of the task are provided. (Bottom) The robot executes the refined behavior and avoids the obstacle.

controllers to allow a human operator to incrementally demonstrate a DMP trajectory safely interacting with the robot during the execution. The limitation of these approaches is that they rely on DMPs that are time dependent DS. As quantitatively shown in [15], time dependent DS use task dependent heuristics to preserve the shape of a demonstrated motion in the face of temporal or spacial perturbations. On the other hand, time-independent DS naturally adapt to motion perturbations and exhibit higher generalization capabilities.

In [12], authors propose to reshape the velocity of an autonomous (i.e. time independent) DS using a modulation matrix. The modulation matrix is parametrized as a rotation matrix and a scalar gain. These parameters are incrementally learned from demonstration using Gaussian processes regression [13]. The approach in [12] has the advantage to locally modify the DS dynamics, i.e. the dynamics in regions of the state space far from the demonstrated trajectories remain unchanged. The main limitations of [12] are that it directly applies only to first-order DS and to low-dimensional spaces (up to 3 dimensions where rotations can be uniquely represented). In our previous work [4], we propose to suppress a learned reshaping term (additive control input) using a time-dependent signal. The reshaping term depends on less parameters than the modulation matrix in [12] and it naturally applies to high-dimensional DS and state spaces. Moreover, generated trajectories accurately follow the demonstrated ones. However, the time-dependence introduces practical limitations due to the hand tuning of the time constant used to suppress the reshaping term. For instance, if a longer trajectory is required for some initial conditions, the reshaping term can be suppressed too early introducing deviations from the demonstrated path.

In this paper, we propose a novel parameterization of the reshaping term that preserves the stability of the reshaped DS without introducing time-dependencies. This is obtained by projecting the reshaping term in the subspace orthogonal

to the gradient of a given Lyapunov function. The presented parameterization is general and no restrictions are imposed on the order of the DS, as well as on the dimension of the state space. Moreover, the dynamics of the original DS are locally affected by the reshaping action, giving the possibility to learn different behaviors in different regions of the state space. To this end, we adopt a kernel based (local) regression technique, namely Gaussian process regression, to retrieve a smooth control input for each state. The control action is learned incrementally from user demonstrations by deciding when new points are added to the training set. As in [4, 12], a trajectory-based sparsity criterion is used to reduce the amount of points added to the training set and reduce the computation time. The incremental learning procedure proposed in this work is shown in Fig. 1.

The rest of the paper is organized as follows. Section 2 presents the theoretical background and the proposed parameterization of the reshaping term. The incremental learning algorithm is described in Section 3. Simulations and experiments are presented in Sec. 4. Section 5 states the conclusions and the future extensions.

2 Orthogonal Reshaping of Dynamical Systems

In this section, we describe an approach to reshape the dynamics of a generic DS without modifying its stability properties.

2.1 Theoretical Background

We assume that the robot's task is encoded into a m-th order autonomous DS (time-dependencies are omitted to simplify the notation)

$$\boldsymbol{p}^{(m)} = \boldsymbol{g}\left(\boldsymbol{p}, \boldsymbol{p}^{(1)}, \ldots, \boldsymbol{p}^{(m-1)}\right) \tag{1}$$

where $\boldsymbol{p} \in \mathbb{R}^n$ is the robot position (in joint or Cartesian space), $\boldsymbol{p}^{(i)}$ is the i-th time derivative of $\boldsymbol{p}$ and $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ is, in general, a non-linear function. The m-th order dynamics (1) can be rewritten, through the change of variables $\boldsymbol{x}^T = [\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_m^T] = [\boldsymbol{p}^T, \ldots, (\boldsymbol{p}^{(m-1)})^T]$, as the first-order dynamics

$$\begin{cases} \dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2 \\ \quad \cdots \\ \dot{\boldsymbol{x}}_m = \boldsymbol{g}\left(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\right) \end{cases} \longrightarrow \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^{mn}$ is the state vector. The solution of (2) $\boldsymbol{\Phi}(\boldsymbol{x}_0, t) \in \mathbb{R}^{mn}$ is called trajectory. Different initial conditions $\boldsymbol{x}_0$ generate different trajectories.

A point $\hat{\boldsymbol{x}} : \boldsymbol{f}(\hat{\boldsymbol{x}}) = \boldsymbol{0} \in \mathbb{R}^{mn}$ is an equilibrium point. An equilibrium is locally asymptotically stable (LAS) if $\lim_{t\to+\infty} \boldsymbol{\Phi}(\boldsymbol{x}_0, t) = \hat{\boldsymbol{x}}, \forall \boldsymbol{x}_0 \in S \subset \mathbb{R}^{mn}$. If $S = \mathbb{R}^{mn}$, $\hat{\boldsymbol{x}}$ is globally asymptotically stable (GAS) and it is the only equilibrium of the DS. A sufficient condition for $\hat{\boldsymbol{x}}$ to be GAS is that there exists a scalar,

continuously differentiable function of the state variables $V(\boldsymbol{x}) \in \mathbb{R}$ satisfying (3a)-(3c) (see, for example, [11] for further details).

$$V(\boldsymbol{x}) \geq 0,\ \forall \boldsymbol{x} \in \mathbb{R}^{mn} \text{ and } V(\boldsymbol{x}) = 0 \iff \boldsymbol{x} = \hat{\boldsymbol{x}} \tag{3a}$$

$$\dot{V}(\boldsymbol{x}) \leq 0,\ \forall \boldsymbol{x} \in \mathbb{R}^{mn} \text{ and } \dot{V}(\boldsymbol{x}) = 0 \iff \boldsymbol{x} = \hat{\boldsymbol{x}} \tag{3b}$$

$$V(\boldsymbol{x}) \to \infty \text{ as } \|\boldsymbol{x}\| \to \infty \text{ (radially unbounded)} \tag{3c}$$

Note that, if condition (3c) is not satisfied, the equilibrium point is LAS. A function satisfying (3a)-(3b) is called a Lyapunov function.

2.2 Reshaping Control Input

If the task consists in reaching a specific position $\hat{\boldsymbol{x}}$ (discrete movement), one can assume that (2) has a GAS equilibrium at $\hat{\boldsymbol{x}}$ and that a Lyapunov function $V(\boldsymbol{x})$ is known [6, 14]. Let us consider the reshaped DS in the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x}) \tag{4}$$

where $\boldsymbol{u}(\boldsymbol{x}) = [\boldsymbol{0}, \dots, \boldsymbol{0}, \boldsymbol{u}_m(\boldsymbol{x})] \in \mathbb{R}^{mn}$ is a continuous control input that satisfies

$$\boldsymbol{u}(\hat{\boldsymbol{x}}) = \boldsymbol{0} \iff \boldsymbol{u}_m(\hat{\boldsymbol{x}}) = \boldsymbol{0} \in \mathbb{R}^n \tag{5a}$$

$$V_x \boldsymbol{u}(\boldsymbol{x}) = 0 \iff V_{x_m} \boldsymbol{u}_m(\boldsymbol{x}) = \frac{\partial V}{\partial \boldsymbol{x}_m} \boldsymbol{u}_m(\boldsymbol{x}) = 0, \forall \boldsymbol{x}_m \in \mathbb{R}^n \tag{5b}$$

where V_x indicates the gradient of $V(\boldsymbol{x})$ with respect to $\boldsymbol{x}$, i.e. $V_x = \partial V(\boldsymbol{x})/\partial \boldsymbol{x}$.

Under conditions (5) the following theorem holds:

Theorem 1. *A GAS equilibrium $\hat{\boldsymbol{x}}$ of (2) is also a GAS equilibrium of the reshaped DS (4).*

Proof. From (4) and (5a) it holds that $\boldsymbol{f}(\hat{\boldsymbol{x}}) + \boldsymbol{u}(\hat{\boldsymbol{x}}) = \boldsymbol{0}$, i.e. $\hat{\boldsymbol{x}}$ is an equilibrium of (4). To analyze the stability of $\hat{\boldsymbol{x}}$, let us consider $V(\boldsymbol{x})$, the Lyapunov function for (2), as a candidate Lyapunov function for (4). Being $V(\boldsymbol{x})$ a Lyapunov function for (2) it satisfies conditions (3a) and (3c) also for the reshaped DS (4). The condition (3b) can expressed in terms of the gradient of $V(\boldsymbol{x})$ as

$$\begin{aligned}\dot{V}(\boldsymbol{x}) = V_x \dot{\boldsymbol{x}} &= \left[\frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{x}_1}, \dots, \frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{x}_m}\right] (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x})) \\ &= \frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{x}_1} \boldsymbol{x}_2 + \dots + \frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{x}_m} \boldsymbol{g}(\boldsymbol{x}) + \cancel{\frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{x}_m} \boldsymbol{u}_m(\boldsymbol{x})} = V_x \boldsymbol{f}(\boldsymbol{x}) < 0, \forall \boldsymbol{x} \neq \hat{\boldsymbol{x}}\end{aligned}$$

where $(\partial V(\boldsymbol{x})/\partial \boldsymbol{x}_m)\boldsymbol{u}_m(\boldsymbol{x}) = 0$ by assumption (5b).

Corollary 1. *Theorem 1 implies that $\hat{\boldsymbol{x}}$ is the only equilibrium point of (4), i.e. $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x})$ vanishes only at $\boldsymbol{x} = \hat{\boldsymbol{x}}$.*

Proof. In Theorem 1 it is proved that $V_x(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x}))$ vanishes only at $\hat{\boldsymbol{x}}$. Hence, also $\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x})$ vanishes only at $\hat{\boldsymbol{x}}$.

Corollary 2. *Theorem 1 still holds for a LAS equilibrium, i.e. if $\hat{\boldsymbol{x}}$ is a LAS equilibrium of (2) then $\hat{\boldsymbol{x}}$ is a LAS equilibrium of (4).*

Proof. If $\hat{\boldsymbol{x}}$ is LAS, only the conditions (3a)-(3b) are satisfied $\forall \boldsymbol{x} \in S \subset \mathbb{R}^{mn}$. The proof of Theorem (1) still holds if $\boldsymbol{x} \in S \subset \mathbb{R}^{mn}$.

Theorem 1 has a clear physical interpretation for second-order dynamical systems in the form $\dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2$, $\dot{\boldsymbol{x}}_2 = \boldsymbol{K}(\hat{\boldsymbol{x}}_1 - \boldsymbol{x}_1) - \boldsymbol{D}\boldsymbol{x}_2$, where $\boldsymbol{K}$ and $\boldsymbol{D}$ are positive definite matrices and $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{x}}_1^T, \boldsymbol{0}^T]^T$ is the equilibrium point. The GAS of $\hat{\boldsymbol{x}}$ can be proven through the energy-based Lyapunov function $V = \frac{1}{2}(\hat{\boldsymbol{x}}_1 - \boldsymbol{x}_1)^T \boldsymbol{K}(\hat{\boldsymbol{x}}_1 - \boldsymbol{x}_1) + \frac{1}{2}\boldsymbol{x}_2^T\boldsymbol{x}_2$ and the La Salle's theorem [11]. The assumptions on $\boldsymbol{u}_m$ in Theorem 1 can be satisfied by choosing $\boldsymbol{u}_m$ orthogonal to $V_{x_2} = \boldsymbol{x}_2$. Hence, $\boldsymbol{u}_m$ is a force that does no work (orthogonal to the velocity), i.e. $\boldsymbol{u}_m$ modifies the trajectory of the system but not its energy.

2.3 Control input parametrization

In order to satisfy the conditions (5) we choose the control input $\boldsymbol{u}_m$ in (4) as

$$\boldsymbol{u}_m = \begin{cases} \boldsymbol{N}\boldsymbol{u}_d = \boldsymbol{N}h(\boldsymbol{x}_1)\left(\boldsymbol{p}_d(\boldsymbol{x}_1) - \boldsymbol{x}_1\right) & \text{if } \boldsymbol{x} \neq \hat{\boldsymbol{x}} \\ \boldsymbol{0} & \text{otherwise} \end{cases} \tag{6}$$

where $\boldsymbol{x}_1 \in \mathbb{R}^n$ is the position of the robot. The scalar gain $h(\boldsymbol{x}_1) \geq 0$ and the desired position $\boldsymbol{p}_d(\boldsymbol{x}_1) \in \mathbb{R}^n$ are learned from demonstrations (see Sec. 3). The adopted parametrization (6) requires always $n+1$ parameters, i.e. the position vector $\boldsymbol{p}_d \in \mathbb{R}^n$ (where n is the Cartesian or joint space dimension) and the scalar gain $h \in \mathbb{R}$. For comparison, consider that the parametrization in [12] uses a rotation and a scalar gain and that a minimal representation of the orientation in $\mathbb{R}^n$ requires at least $n(n-1)/2$ parameters [24]. The vector $\boldsymbol{u}_d$ represents an elastic force attracting the position $\boldsymbol{x}_1$ towards the desired position $\boldsymbol{p}_d$. The matrix $\boldsymbol{N}$ is used to project $\boldsymbol{u}_d$ into the subspace orthogonal to V_{x_m} and it is defined as

$$\boldsymbol{N} = \|V_{x_m}\|^2 \left(\boldsymbol{I}_{n\times n} - \bar{V}_{x_m}^T \bar{V}_{x_m}\right) \tag{7}$$

where $\boldsymbol{I}_{n\times n}$ is the n-dimensional identity matrix and $\bar{V}_{x_m} = V_{x_m}/\|V_{x_m}\|$. The term $\|V_{x_m}\|^2$ guarantees a smooth convergence of $\boldsymbol{u}_m$ to zero. Note that the control input $\boldsymbol{u}_m \to \boldsymbol{0}$ if $h(\boldsymbol{x}_1) \to 0$. This property is exploited in Sec. 3 to locally modify the trajectory of the DS.

3 Learning Reshaping Terms

In this section, an approach is described to learn and online retrieve for each position $\boldsymbol{x}_1$ the parameter vector $\boldsymbol{\lambda} = [h,\ \boldsymbol{p}_d^T]$ that parametrizes the control input in (6). We use a local regression technique, namely Gaussian process regression (GPR), to ensure that $\boldsymbol{u}_m \to \boldsymbol{0}$ ($h \to 0$) when the robot is far from the demonstrated trajectories. This makes it possible to locally follow the demonstrated trajectories, leaving the rest almost unchanged.

3.1 Compute Training Data

Consider that a new demonstration of a task is given as $\boldsymbol{X} = \{\boldsymbol{x}_{d,1}^t,\ \dot{\boldsymbol{x}}_{d,m}^t\}_{t=1}^T$, where $\boldsymbol{x}_{d,1}^t \in \mathbb{R}^n$ is the desired position at time t and $\dot{\boldsymbol{x}}_{d,m}^t \in \mathbb{R}^n$ is the time derivative of the last state component $\boldsymbol{x}_{d,m}^t$ at time t. For example, if one wants to reshape a second-order DS, then $\boldsymbol{X}$ contains T positions and T accelerations. The procedure to transform the demonstration into T observations of $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}^t = [\lambda_1, \dots, \lambda_{n+1}]^t = \left[h,\ \boldsymbol{p}_d^T\right]^t\}_{t=1}^T$ requires following steps:

I. Set $\{\boldsymbol{p}_d^t = \boldsymbol{x}_{d,1}^t\}_{t=1}^T$, where $\boldsymbol{x}_{d,1}$ are the demonstrated positions.

The gain h^t in (6) multiplies the position error $\boldsymbol{p}_d - \boldsymbol{x}_1$ and it is used to modulate the control action $\boldsymbol{u}_m$ and to improve the overall tracking performance. The value of h^t is computed by considering that $\dot{\boldsymbol{x}}_m = \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{u}_m$ from (2) and (4). The following steps are needed:

II. Create the initial state vector $\boldsymbol{x}_d^0 = [(\boldsymbol{x}_{d,1}^1)^T, \boldsymbol{0}^T, \dots, \boldsymbol{0}^T]^T \in \mathbb{R}^{mn}$.
III. Compute $\left\{\boldsymbol{x}_o^t = \boldsymbol{\Phi}(\boldsymbol{x}_d^0, (t-1)\delta t)\right\}_{t=1}^T$, where $\boldsymbol{\Phi}$ is the solution of (2) with initial condition $\boldsymbol{x}_d^0$ (see Sec. 2.1) and δt is the sampling time.
IV. Compute $\{\boldsymbol{u}_m^t\}_{t=1}^T$ from (6) with $h = 1$ and $\{\boldsymbol{x}_1^t = \boldsymbol{x}_{o,1}^t\}_{t=1}^T$.
V. Set

$$h^t = \begin{cases} \dfrac{\|\dot{\boldsymbol{x}}_{d,m}^t - \boldsymbol{g}(\boldsymbol{x}_o^t)\|}{\|\boldsymbol{u}_m^t\|} & \text{if } \|\boldsymbol{u}_m^t\| > 0 \\ 0 & \text{otherwise} \end{cases} \quad t = 1, \dots, T \tag{8}$$

Once the observations of $\boldsymbol{\lambda}$ are computed, any local regression technique can be applied to learn the relationship between $\boldsymbol{\lambda}$ and the position $\boldsymbol{x}_1$ of the DS to reshape.

3.2 Gaussian Process Regression

Gaussian processes (GP) are widely used to learn input-output mappings from observations [13]. GP models the scalar noisy process $\lambda^t = f(\boldsymbol{x}_1^t) + \epsilon \in \mathbb{R}, t = 1, \dots, T$ with ϵ a Gaussian noise with zero mean and variance σ_n^2. Therefore, n processes λ_i^t are assumed to generate the training input $\boldsymbol{X} = \{\boldsymbol{x}_1^t\}_{t=1}^T$ and output $\boldsymbol{\Lambda}_i = \{\lambda_i^t\}_{t=1}^T$. Given the training pairs $(\boldsymbol{X}, \boldsymbol{\Lambda}_i)$ and a query point $\boldsymbol{x}^*$, it is possible to compute the joint distribution

$$\begin{bmatrix} \boldsymbol{\Lambda}_i \\ \lambda_i^* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{XX} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}_{x^*X} \\ \boldsymbol{K}_{Xx^*} & k(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix}\right) \tag{9}$$

where λ_i^* is the expected output at $\boldsymbol{x}^*$. The matrix $\boldsymbol{K}_{x^*X} = \{k(\boldsymbol{x}^*, \boldsymbol{x}_1^t)\}_{t=1}^T$, $\boldsymbol{K}_{Xx^*} = \boldsymbol{K}_{x^*X}^T$. Each element ij of $\boldsymbol{K}_{XX}$ is given by $\{\boldsymbol{K}_{XX}\}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k(\bullet, \bullet)$ is a user-defined covariance function. In this work, we used the squared exponential covariance function

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_k^2 \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2l}\right) + \sigma_n^2 \delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{10}$$

$k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is parameterized by the 3 positive parameters σ_k^2, σ_n^2, and l. The tunable parameters σ_k^2, σ_n^2, and l can be hand-crafted or learned from training data [13]. We decide to keep them fixed in order to perform incremental learning by simply adding new points to the training set. It is worth noticing that the adopted kernel function (10) guarantees that $\boldsymbol{\lambda} \to \mathbf{0}$ for points far from the demonstrated positions.

Predictions with a GP model are made using the conditional distribution of $\lambda_i^* | \boldsymbol{\Lambda}_i$, i.e.

$$\lambda_i^* | \boldsymbol{\Lambda}_i \sim \mathcal{N}\left(\mu_{\lambda_i^* | \boldsymbol{\Lambda}_i}, \sigma^2_{\lambda_i^* | \boldsymbol{\Lambda}_i}\right) \tag{11}$$

where

$$\begin{aligned} \mu_{\lambda_i^* | \boldsymbol{\Lambda}_i} &= \boldsymbol{K}_{x^* X} \left(\boldsymbol{K}_{XX} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{\Lambda}_i \\ \sigma^2_{\lambda_i^* | \boldsymbol{\Lambda}_i} &= k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{K}_{x^* X} \left(\boldsymbol{K}_{XX} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \boldsymbol{K}_{X x^*} \end{aligned} \tag{12}$$

The mean $\mu_{\lambda_i^* | \boldsymbol{\Lambda}_i}$ approximates λ_i^*, while the variance $\sigma^2_{\lambda_i^* | \boldsymbol{\Lambda}_i}$ plays the role of a confidence bound. If, as in this work, a multidimensional output is considered, one can simply train one GP for each dimension.

To reduce the computation effort due to the matrix inversion in (12), incremental GP algorithms introduce criteria to sparsely represent incoming data [22]. Assuming that T data $\{\boldsymbol{x}_1^t,\ h^t,\ \boldsymbol{p}_d^t\}_{t=1}^T$ are already in the training set, we add a new data point $[\boldsymbol{x}_1^{T+1},\ h^{T+1},\ \boldsymbol{p}_d^{T+1}]$ if the cost

$$C^{T+1} = \|\boldsymbol{p}_d^{T+1} - \hat{\boldsymbol{p}}_d^{T+1}\| > \bar{c} \tag{13}$$

where $\hat{\boldsymbol{p}}_d^{T+1}$ indicates the position predicted at $\boldsymbol{x}_1^{T+1}$ with (12) using only data $\{\boldsymbol{x}_1^t,\ h^t,\ \boldsymbol{p}_d^t\}_{t=1}^T$ already in the training set. Similarly to [4, 12], the tunable parameter $\bar{c}$ represents the error in approximating demonstrated positions and it can be easily tuned. For example, $\bar{c} = 0.2$ means that position errors smaller than 0.2 meters are acceptable. The proposed incremental reshaping approach is summarized in Tab. 1.

4 Results

4.1 Simulation - Learning Bi-Modal Behaviors

The goal of this simulation is to illustrate the incremental nature of the proposed reshaping approach, its ability to learn different behaviors in different regions of the space, and the possibility to reshape high order DS. The original trajectory is obtained by numerically integrating ($\delta t = 0.01\,\mathrm{s}$) the second-order DS

$$\begin{aligned} \dot{\boldsymbol{x}}_1 &= \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_2 &= -10\boldsymbol{x}_1 - 2\sqrt{10}\boldsymbol{x}_2 \end{aligned} \tag{14}$$

Table 1. Proposed reshaping approach.

Batch	Create a set of predefined tasks encoded as stable DS. Stable DS can be designed by the user or learned from demonstrations as in [6, 14].
	Provide a Lyapunov function $V(\boldsymbol{x})$ for each DS.
Incremental	Observe the robot's behavior in novel scenarios.
	If needed, provide a corrective demonstration, for example by kinesthetic teaching the robot.
	Learn the parameters of the control input (4), as described in Sec. 3. Tuning parameters can be set empirically by simulating the reshaped DS.
	Repeat until the refined behavior is satisfactory.

where $\boldsymbol{x}_1 = [x, y]^T \in \mathbb{R}^2$ is the position and $\boldsymbol{x}_2$ the velocity. The system (14) has a GAS equilibrium at $\hat{\boldsymbol{x}} = \mathbf{0} \in \mathbb{R}^4$ and Lyapunov function $V(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x}_1^T \boldsymbol{x}_1 + \boldsymbol{x}_2^T \boldsymbol{x}_2)$.

Local demonstrations are drawn from different Gaussian distributions, as described in Tab. 2, to obtain different bi-modal behaviors. A total of four demonstrations are used in each case, i.e. two (red and green crosses in Fig. 2) for the behavior in the region $\mathcal{R}^+$ where $x > 0$, two (magenta and blue crosses in Fig. 2) for the region $\mathcal{R}^-$ where $x < 0$. As shown in Fig. 2, the original DS position trajectory (black solid line) is incrementally adapted to follow the demonstrated positions and different behaviors are effectively learned in $\mathcal{R}^+$ and $\mathcal{R}^-$. Totally four simulations are conducted and shown in Fig. 2. In all the presented cases, the DS is successfully reshaped to follow the demonstrated trajectories. The proposed approach locally modifies the DS, in fact demonstrations in $\mathcal{R}^+$, being far from $\mathcal{R}^-$, do not affect the behavior in $\mathcal{R}^-$ (and vice versa). The equilibrium position $\hat{\boldsymbol{x}}_1 = \mathbf{0} \in \mathbb{R}^2$ is always reached, an expected result considering Theorem 1. Results are obtained with noise variance $\sigma_n^2 = 0.1$, signal variance $\sigma_k^2 = 1$, length scale $l = 0.4$, and threshold $\bar{c} = 0.02\,\text{m}$.

4.2 Experiments

The effectiveness of the proposed approach is demonstrated with two experiments on a KUKA LWR IV+ 7 DoF manipulator. In both experiments, novel demonstrations of the desired position are provided to the robot by kinesthetic teaching. To guarantee a safe physical guidance, the task is interrupted and the robot is put in the gravity compensation mode as soon as the user touches the robot. The external torques estimation provided by the fast research interface [**?**] is used to detect physical contacts.

Table 2. Demonstrations used for the four simulations in Fig. 2.

Figure	Demonstrations in $\mathcal{R}^-$	Demonstrations in $\mathcal{R}^+$
2(a)	$x \in [-2, -2.5]$ $y = 0.1\mathcal{N}(-2, 0.03)$	$x \in [2, 2.5]$ $y = 0.1\mathcal{N}(2, 0.03)$
2(b)	$x \in [-2, -2.5]$ $y = -0.1\mathcal{N}(-2, 0.03)$	$x \in [2, 2.5]$ $y = 0.1\mathcal{N}(2, 0.03)$
2(c)	$x \in [-2, -2.5]$ $y = 0.1\mathcal{N}(-2, 0.03)$	$x \in [2, 2.5]$ $y = -0.1\mathcal{N}(2, 0.03)$
2(d)	$x \in [-2, -2.5]$ $y = -0.1\mathcal{N}(-2, 0.03)$	$x \in [2, 2.5]$ $y = -0.1\mathcal{N}(2, 0.03)$

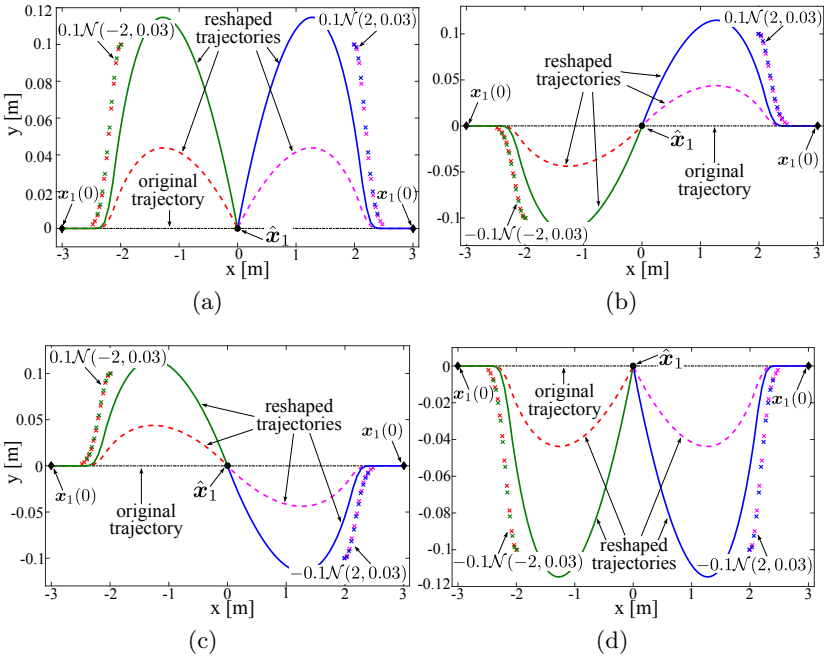


Fig. 2. Different bi-modal behaviors obtained by reshaping the same dynamical system. Red and magenta dashed lines are the reshaped trajectories after providing two demonstrations, one (red crosses) for $\mathcal{R}^+$ and one (magenta crosses) for $\mathcal{R}^-$. Blue and green solid lines are the reshaped trajectories after providing four demonstrations.

End-effector Collision Avoidance This experiment shows the ability of the proposed reshaping approach to learn different behaviors in different regions of the space and the possibility to reshape non-linear DS. The task consists in reaching the goal position $\hat{\boldsymbol{x}}_1 = [-0.52, 0, 0.02]^T$ m with the robot's end-effector while avoiding a box (see Fig. 3(d)) of size $7 \times 7 \times 23$ cm. Two boxes are placed

in the scene in different positions, one in the region $\mathcal{R}^+$ where $y > 0$ (Fig. 3(d)), one in the region $\mathcal{R}^-$ where $y < 0$ (Fig. 3(e)). Hence, the robot has to learn a bi-modal behavior to avoid collisions in $\mathcal{R}^+$ and $\mathcal{R}^-$.

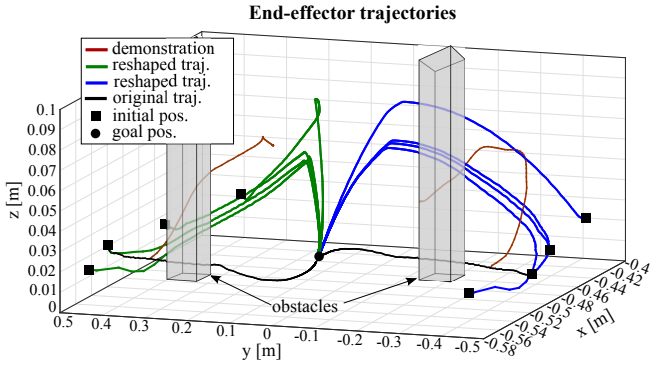
The original position trajectory is obtained by numerically integrating ($\delta t = 0.005\,\text{s}$) the first-order and non-linear DS $\dot{\boldsymbol{x}}_1 = \boldsymbol{f}(\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1)$, where $\boldsymbol{x}_1 = [x, y, z]^T \in \mathbb{R}^3$ is the end-effector position and $\dot{\boldsymbol{x}}_1 \in \mathbb{R}^3$ is the end-effector linear velocity. The orientation is kept fixed. The original DS is learned from demonstrations by using the approach in [14]. The Lyapunov function for the original DS is $V = \frac{1}{2}(\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1)^T(\boldsymbol{x}_1 - \hat{\boldsymbol{x}}_1)$ [14]. The original end-effector trajectories are shown in Fig. 3(a)–(c) (black solid lines).

Following the original trajectory generated with initial position $\boldsymbol{x}_1(0) = [-0.52, 0.5, 0.02]^T$ m (or $\boldsymbol{x}_1(0) = [-0.52, -0.5, 0.02]^T$ m), the robot hits the box. To prevent this, two partial demonstrations (one in $\mathcal{R}^+$ and one $\mathcal{R}^-$) are provided to show to the robot how to avoid the obstacles (brown solid lines in Fig. 3(a)–(c)). The original DS position trajectories (black solid lines in Fig. 3(a)–(c)) are incrementally adapted to follow the demonstrated positions and different avoiding behaviors are effectively learned in $\mathcal{R}^+$ and $\mathcal{R}^-$.

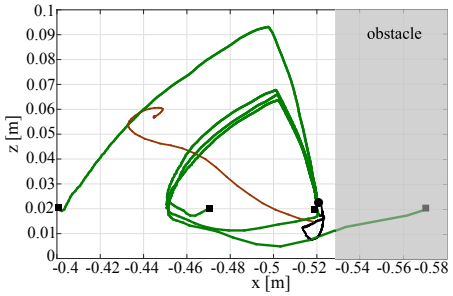
The proposed approach locally modifies the DS, in fact demonstrations in $\mathcal{R}^+$, being far from $\mathcal{R}^-$, do not affect the behavior in $\mathcal{R}^-$ (and vice versa). The equilibrium position $\hat{\boldsymbol{x}}_1 = [-0.52, 0, 0.02]^T$ m is always reached, as stated by Theorem 1. Figure 3 also shows the learned behaviors (green in $\mathcal{R}^+$ and blue in $\mathcal{R}^-$ solid lines) for different initial positions in a 3D view (Fig. 3(a)) and in the xz plane (Fig. 3(b) and 3(c)). In all cases, the robot is able to achieve the task. Snapshots of the learned bi-modal behavior are depicted in Fig. 3(d) and 3(e). Results are obtained with noise variance $\sigma_n^2 = 0.1$, signal variance $\sigma_k^2 = 1$, length scale $l = 0.001$, and the threshold $\bar{c} = 0.04\,\text{m}$. With the adopted $\bar{c}$ only 106 points over 798 are added to the GP.

Joint Space Collision Avoidance This experiment shows the scalability of the proposed approach to high dimensional spaces and its ability to reshape high order DS. The task is a point-to-point motion in the joint space from $\boldsymbol{x}_1(0) = [35, 55, 15, -65, -15, 50, 90]^T$ deg to $\hat{\boldsymbol{x}}_1 = [-60, 30, 30, -70, -15, 85, 15]^T$ deg. The original joint position trajectory is obtained by numerically integrating ($\delta t = 0.005\,\text{s}$) the second-order DS $\dot{\boldsymbol{x}}_1 = \boldsymbol{x}_2$, $\dot{\boldsymbol{x}}_2 = 2(\hat{\boldsymbol{x}}_1 - \boldsymbol{x}_1) - 2\sqrt{2}\boldsymbol{x}_2$, where $\boldsymbol{x}_1 = [\theta_1, \dots, \theta_7]^T \in \mathbb{R}^7$ are the joint angles and $\boldsymbol{x}_2 \in \mathbb{R}^7$ the joint velocities. The system has a GAS equilibrium at $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{x}}_1^T, \boldsymbol{0}^T]^T \in \mathbb{R}^{14}$ and Lyapunov function $V = \frac{1}{2}(\hat{\boldsymbol{x}}_1 - \boldsymbol{x}_1)^T(\hat{\boldsymbol{x}}_1 - \boldsymbol{x}_1) + \frac{1}{2}\boldsymbol{x}_2^T\boldsymbol{x}_2$. The original joint angle trajectories are shown in Fig. 4 (black solid lines).

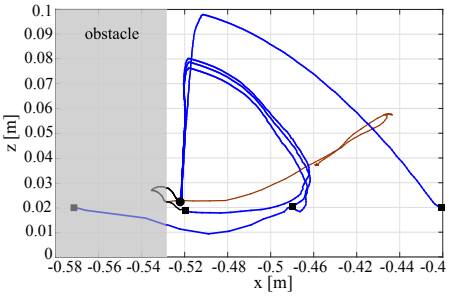
As shown in Fig. 1, following the original trajectory the robot hits an unforeseen obstacle (the red bar in Fig. 1). A kinesthetic demonstration is then provided (red lines in Fig. 4) to avoid the collision. With the reshaped trajectory (blue lines in Fig. 4) the robot is able to avoid the obstacle (Fig. 1) and to reach the desired goal $\hat{\boldsymbol{x}}_1$. Results are obtained with noise variance $\sigma_n^2 = 0.1$, signal variance $\sigma_k^2 = 1$, length scale $l = 0.01$, and the threshold $\bar{c} = 15\,\text{deg}$. With the adopted $\bar{c}$ only 27 points over 178 are added to the GP.



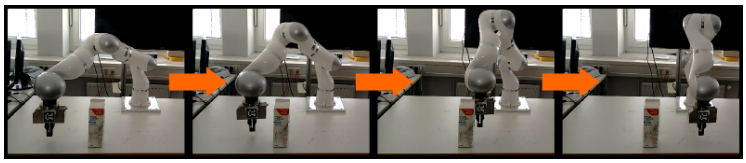
(a) 3D view.



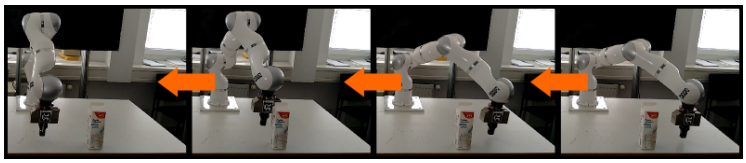
(b) XZ view in $\mathcal{R}^+$.



(c) XZ view in $\mathcal{R}^-$.



(d) Reshaped trajectory starting at $\boldsymbol{x}_1(0) \in \mathcal{R}^+$.



(e) Reshaped trajectory starting at $\boldsymbol{x}_1(0) \in \mathcal{R}^-$.

Fig. 3. Results of the end-effector collision avoidance experiment.

4.3 Discussion

The proposed approach works for high order DS, as underlined in Sec. 2 and demonstrated in Sec. 4.2. Being robot manipulators dynamics described by second-order DS, a second-order DS is sufficient to generate dynamically feasible

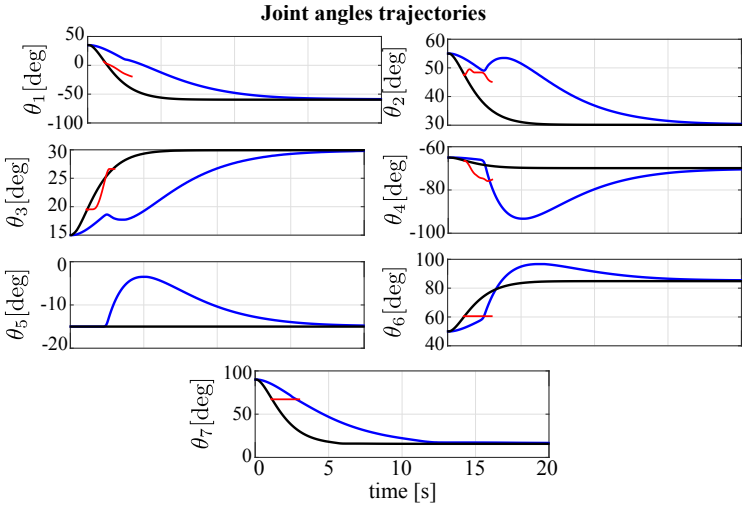


Fig. 4. Original joint angles trajectories (black lines), the provided demonstration (red lines) and reshaped joint angles trajectories (blue lines) for the joint space collision avoidance experiment.

trajectories. For this reason, we show results for DS up to the second order. The adopted control law in (6) pushes the robot position towards the demonstrated position, without considering desired velocities or accelerations. We adopt this solution because, in the majority of the cases, a user is interested in reconfiguring the robot and (s)he can hardly show a desired velocity (or acceleration) behavior through kinesthetic teaching.

It must be noted that the proposed control law (6) does not always guarantee good tracking of the demonstrated trajectories, as shown in Fig. 4. In general, to have good tracking performance, different controllers have to be designed for different DS [11]. Nevertheless, in this work we do not focus on accurately tracking the demonstrated trajectories, but we want to modify the robot's behavior until the task is correctly executed. The joint angle trajectories in Fig. 4 guarantee the correct execution of the task, i.e. the robot converges to the desired joint position while avoiding the obstacle. The loss of accuracy also depends on the orthogonality constraint in (5b) between the gradient of the Lyapunov function and the control input. This constrain allows only motions perpendicular to the gradient to be executed, which limits the control capabilities and increases the number of demonstrations needed in order to obtain the satisfactory behavior. In principle, it is possible to relax the constraint (5b) by requiring that $V_x \boldsymbol{u}(\boldsymbol{x}) \geq 0$. The design of a control input the satisfies $V_x \boldsymbol{u}(\boldsymbol{x}) \leq 0$ is left as a future work.

Figure 4 shows an overshoots in the resulting position trajectory (see, for instance, the angle θ_6). To better understand this behavior, consider that we are controlling a spring-damper (linear) DS with a proportional controller (with a non-linear gain). In case the resulting closed-loop system is not critically damped,

the retrieved trajectory overshoots the goal position. For the experiment in Sec. 4.2, adding a damping control action (PD-like controller) would solve the overshoot problem. However, there is no guarantee that a generic non-linear DS does not overshoot under a PD-like control action. Moreover, adding another term to (6) will increase the number of parameters to learn. Therefore, we use a proportional controller in this work and leave the learning of more sophisticated controllers as a future extension.

5 Conclusions and Future Work

We presented a novel approach to incrementally modify the position trajectory of a generic dynamical system, useful to on-line adapt predefined tasks to different scenarios. Compared to state-of-the-art approaches, our method works also for high-order dynamical systems, preserves the time-independence of the DS, and does not affect the stability properties of the reshaped dynamical system, as shown in the conducted Lyapunov-based stability analysis.

A control law is proposed that locally modifies the trajectory of the dynamical system to follow a desired position. Desired positions, as well as the control gain, are 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 learned behavior is not satisfactory. Due to the local nature of the reshaping control input, different behaviors can be learned and executed in different regions of the space. Simulations and experiments on a 7 DoF manipulator show the effectiveness of the proposed approach in reshaping non-linear, high-order dynamical systems, and its scalability to high dimensional spaces (up to $\mathbb{R}^{14}$).

Our approach applies to dynamical systems with a LAS or a GAS equilibrium point. Nevertheless, DS that converges towards periodic orbits (limit cycles) have been used in robotic applications to generate periodic behaviors [6]. Compared to static equilibria, limit cycles stability has a different characterizations in terms of Lyapunov analysis. Our next research will focus on considering incremental reshaping of periodic motions while preserving their stability properties.

References

1. Saveriano, M., Lee, D.: Point Cloud based Dynamical System Modulation for Reactive Avoidance of Convex and Concave Obstacles. International Conference on Intelligent Robots and Systems, pp. 5380–5387 (2013).
2. Saveriano, M., Lee, D.: Distance based Dynamical System Modulation for Reactive Avoidance of Moving Obstacles. International Conference on Robotics and Automation, pp. 5618–5623 (2014).
3. Blocher, C., Saveriano, M., Lee, D.: Learning Stable Dynamical Systems using Contraction Theory. Ubiquitous Robots and Ambient Intelligence, pp. 124–129 (2017).
4. Saveriano, M., Lee, D.: Incremental Skill Learning of Stable Dynamical Systems. International Conference on Intelligent Robots and Systems, pp. 6574–6581 (2018).

5. Saveriano, M., Hirt, F., Lee, D.: Human-aware motion reshaping using dynamical systems. Pattern Recognition Letters 99, 96–104 (2017).
6. Ijspeert, A., Nakanishi, J., Pastor, P., Hoffmann, H., Schaal, S.: Dynamical Movement Primitives: learning attractor models for motor behaviors. Neural Computation 25:2, 328–373 (2013).
7. Khansari-Zadeh, S. M., Billard, A.: A dynamical system approach to realtime obstacle avoidance. Autonomous Robots 32:4, 433–454 (2012).
8. Karlsson, M., Robertsson, A., Johansson, R.: Autonomous interpretation of demonstrations for modification of dynamical movement primitives. International Conference on Robotics and Automation, pp. 316–321 (2017).
9. Talignani Landi, C., Ferraguti, F., Fantuzzi, C., Secchi, C.: A passivity-based strategy for coaching in human–robot interaction. International Conference on Robotics and Automation, pp. 3279–3284 (2018).
10. Kastritsi, T., Dimeas, F., Doulgeri, Z.: Progressive Automation with DMP Synchronization and Variable Stiffness Control. Robotics and Automation Letters 3:4, 3279–3284 (2018).
11. Slotine, J. J. E., Li, W.: Applied nonlinear control. Prentice-Hall (1991).
12. Kronander, K., Khansari-Zadeh, S. M., Billard, A.: Incremental Motion Learning with Locally Modulated Dynamical Systems. Robotics and Autonomous Systems 70, pp. 52–62 (2015).
13. Rasmussen, C. E., Williams, C. K. I.: Incremental Gaussian processes for machine learning. MIT Press (2006).
14. Khansari-Zadeh, S. M., Billard, A.: Learning Stable Non-Linear Dynamical Systems with Gaussian Mixture Models. Transactions on Robotics 27:5, 943–957 (2011).
15. Gribovskaya, E., Khansari-Zadeh, S. M., Billard, A.: Learning Non-linear Multivariate Dynamics of Motion in Robotic Manipulators. The International Journal of Robotics Research 30:1, 80—117 (2011)
16. Saveriano, M., An, S., Lee, D.: Incremental Kinesthetic Teaching of End-Effector and Null-Space Motion Primitives. International Conference on Robotics and Automation, pp. 3570–3575 (2015).
17. Saveriano, M., Franzel, F., Lee, D.: Merging position and orientation motion primitives. International Conference on Robotics and Automation, pp. 7041–7047 (2019).
18. Saveriano, M., Lee, D.: Learning Motion and Impedance Behaviors from Human Demonstrations. International Conference on Ubiquitous Robots and Ambient Intelligence, pp. 368–373 (2014).
19. Lee, D., Ott, C.: Incremental Kinesthetic Teaching of Motion Primitives Using the Motion Refinement Tube. Autonomous Robots 31:2, 115–131 (2011).
20. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot Programming by Demonstration. Springer Handbook of Robotics, pp. 1371–1394 (2008).
21. Calinon, S., Guenter, F., Billard, A.: On Learning, Representing, and Generalizing a Task in a Humanoid Robot. Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37:2, 286–298 (2007).
22. Csató, L.: Gaussian Processes - Iterative Sparse Approximations. PhD Dissertation, Aston University (2002).
23. Calinon, S., Sardellitti, I., Caldwell, D.: The Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. International Conference on Intelligent Robots and Systems, 249–254 (2010).
24. Mortari, D.: On the rigid rotation concept in n-dimensional spaces. Journal of the Astronautical Sciences 49:3, 401–420 (2001).