Learning Control

Sylvain Calinon[*]and Dongheui Lee[†]

Abstract

This chapter presents an overview of learning approaches for the acquisition of controllers and movement skills in humanoid robots. The term learning control refers to the process of acquiring a control strategy to achieve a task. While the definition is in some cases restrained to trial-and-error learning, we present here learning control in a broader perspective, with a focus on the representation of skills to be acquired, and on the different learning strategies that can contribute to the acquisition of robust and adaptive controllers for humanoids.

Keywords: control policy learning, motor skill acquisition, movement primitives, learning from demonstration, skill transfer, reward-weighted optimization, regression

Contents

1 Introduction

Humanoids present unique challenges for learning control. The structural resemblance with humans can be exploited to provide intuitive teaching interactions and to facilitate the mapping between users and robots. Learning controllers in humanoids can leverage upon different modalities and learning strategies, including visual observation, kinesthetic teaching, haptic correction, visual or oral feedback, and self-refinement.

Figure 1 presents examples of skills acquisition with humanoids. These learning challenges are characterized by an exceptional richness of structures and constraints that can be explored with such platforms. When considering humanoid controllers, the number of demonstrations or trials that is considered will often be much lower in comparison to other fields in machine learning. This characteristic provides a unique opportunity to develop models and algorithms dedicated to humanoids, that can learn similarly to humans, and that can generalize control skills to new situations through demonstration and iterative learning. The selection of models and tools used in practice for learning control is often guided by this mismatch between the scarce number of observation/execution trials and the high number of degrees of freedom to sense and control. In terms of encoding, generative models can, for example, be selected when there is the need to both recognize and synthesize controllers. The scarcity of data also tends to favor linear regression techniques. We will see in the chapter that linear regression is often used in one way or another, either locally or in a kernel form, in order to provide nonlinear global behaviors while keeping locally linear behaviors. Since the combination of learning and control is a challenge *per se*, the techniques used in practice might appear as basic techniques from a learning or control theory perspective (often revolving around variants of mixture models or linear quadratic regulators), but have been shown in practice to be efficient. From a modeling perspective, the combination of learning and control techniques can also be facilitated by the correspondence between quadratic error minimization in linear systems and log-likelihood optimization in statistical learning, which is often the key to many approaches combining learning and control.

[*]Idiap Research Institute, Martigny, Switzerland, E-mail: sylvain.calinon@idiap.ch.

[†]Technical University of Munich (TUM), and Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany, E-mail: dhlee@tum.de, dongheui.lee@dlr.de.

The final publication is available at Springer via https://doi.org/10.1007/978-94-007-7194-9_68-1.

The chapter is organized as follows. Section 2 presents different ways of collecting data when learning control policies. Section 3 presents various approaches to represent control policies in a compact and modular manner, by relying on movement primitives and associated regression algorithms. The section covers weighted least squares (§3.1), locally weighted regression (§3.2), dynamical movement primitives (§3.3), Gaussian mixture regression (§3.4), Gaussian process regression (§3.5) as well as various forms of trajectory distributions (§3.6), including ProMP (§3.6.1) and trajectory-GMM (§3.6.2). We then briefly introduce extensions to hidden Markov models (§3.7), followed by the introduction of various forms of autonomous dynamical systems (§3.8).

Section 4 presents the use of these representations in the context of control policy learning. Section 4.1 introduces their use in the context of linear quadratic tracking. Section 4.2 covers optimization strategies based on reward-weighted averaging. Section 4.3 introduces iterative learning control. Section 4.4 extends the control policy learning problem to the task prioritization issue in humanoids. Section 4.5 finally illustrates the exploitation of these tools in application examples.

§5 concludes this chapter by presenting future prospects and open research directions. The source codes of simple didactic examples accompany the different sections of this chapter, and are available at
`http://www.idiap.ch/software/pbdlib/`.

2 Learning modalities for skills acquisition

2.1 Visual observation

Visual observations of human motions can be exploited in imitation learning. Three-dimensional motion capture systems can be used for mimicking the user's motion to a robot, which is particularly convenient in the case of humanoid robots. The overall structure consists of three components: human motion measurement, motion mapping from a human to a robot, and motion control.

Human motions can be measured by different types of systems, such as marker-based optical motion capture, inertial motion capture, markerless optical motion capture, and time of flight sensors. Numerous techniques exist for human motion measurement and many commercial motion capture systems are available on the market, such as Vicon (`http://www.vicon.com`) or Optitrack (`http://www.optitrack.com`).

The motion mapping problem concerns the mapping of human motions to robot motions in spite of their kinematic and dynamic differences. The earlier work in online mapping started with the imitation of upper-body motions. Pollard *et al.* [76] adapted the joint angles of a human skeleton model to a robot according to joint and velocity limits by local scaling. In order to measure quantitatively the humanlikeness of the arm motions, Kim *et al.* [42] defined the elbow elevation angle and solved the inverse kinematics analytically according to the geometric relations by specifying six holonomic constraints for each arm.

Another approach followed by [19, 17, 71, 32] is to develop controllers in task space (i.e., marker space). In [19], different priorities are assigned to the markers and realized based on projections into subsequent nullspaces by using the operational space approach. Dariush *et al.* [17] utilize a closed-loop inverse kinematics strategy. In the Cartesian control method for the retargeting of human motion to a humanoid robot in [71], a set of control points on the humanoid robot is connected to the corresponding measured points on a human via virtual springs, where the spring forces drive a simulation of the robot dynamics. The approach does not require a hard priority order between the markers, and is applicable both for position control and torque control.

Although the imitation of upper-body motion has been explored with both task and joint space mappings at kinematic level, lower-body motions are often either neglected or replaced by simple balance control without considering the motion similarity of the lower body. Online footprint imitation is realized on the MAHRU-R robot by feeding the recognized and adapted human step parameters into a classical zero-moment point (ZMP)-based walking controller [43]. In [32, 46] whole-body human motions are imitated in real time including stance foot changes by a NAO humanoid robot. Foot position and orientation trajectories are faithfully imitated rather than using a predefined foot step plan. In [32, 71], the imitation algorithm is combined with the paradigm of learning from demonstrations. It is shown in [32] that the delay of online human-legged motion imitation could be reduced by learning and predicting human stances. By handling motion similarity both in task space and joint space, more humanlike motions are achieved in [33], namely knee-stretched walking by considering footprints, knee angles, etc. The online walking imitation is formulated as an optimization problem with a set of task space and joint space tracking targets with different priorities, where conflicts in these two spaces were resolved by taking into account dynamic constraints during the different walking phases.

2.2 Kinesthetic teaching

Physical interaction provides a natural interface to kinesthetic transfer of skills, where the user can demonstrate or refine the task in the robot's environment while feeling its capabilities and limitations [10, 38, 55]. The recent hardware and software developments toward compliant and tactile robots (including variable stiffness actuators, backdrivable motors, and artificial skins) make kinesthetic teaching a promising teaching modality for the user.

Early work in kinesthetic teaching considered a passive robot behavior [10, 38] by deactivating the controlled motion or setting very low servo gains. Therein the teacher might tend to move motors one by one rather than demonstrating natural coordinated movements. The development of torque-controlled robots initiated new approaches combining active impedance control and physical teaching [54, 55, 83].

Methods for incremental learning using multiple learning modalities [54, 55] offer intuitive teaching of natural motions and ensure synchronization of complex whole-body motions. Therein, the learning procedure starts with

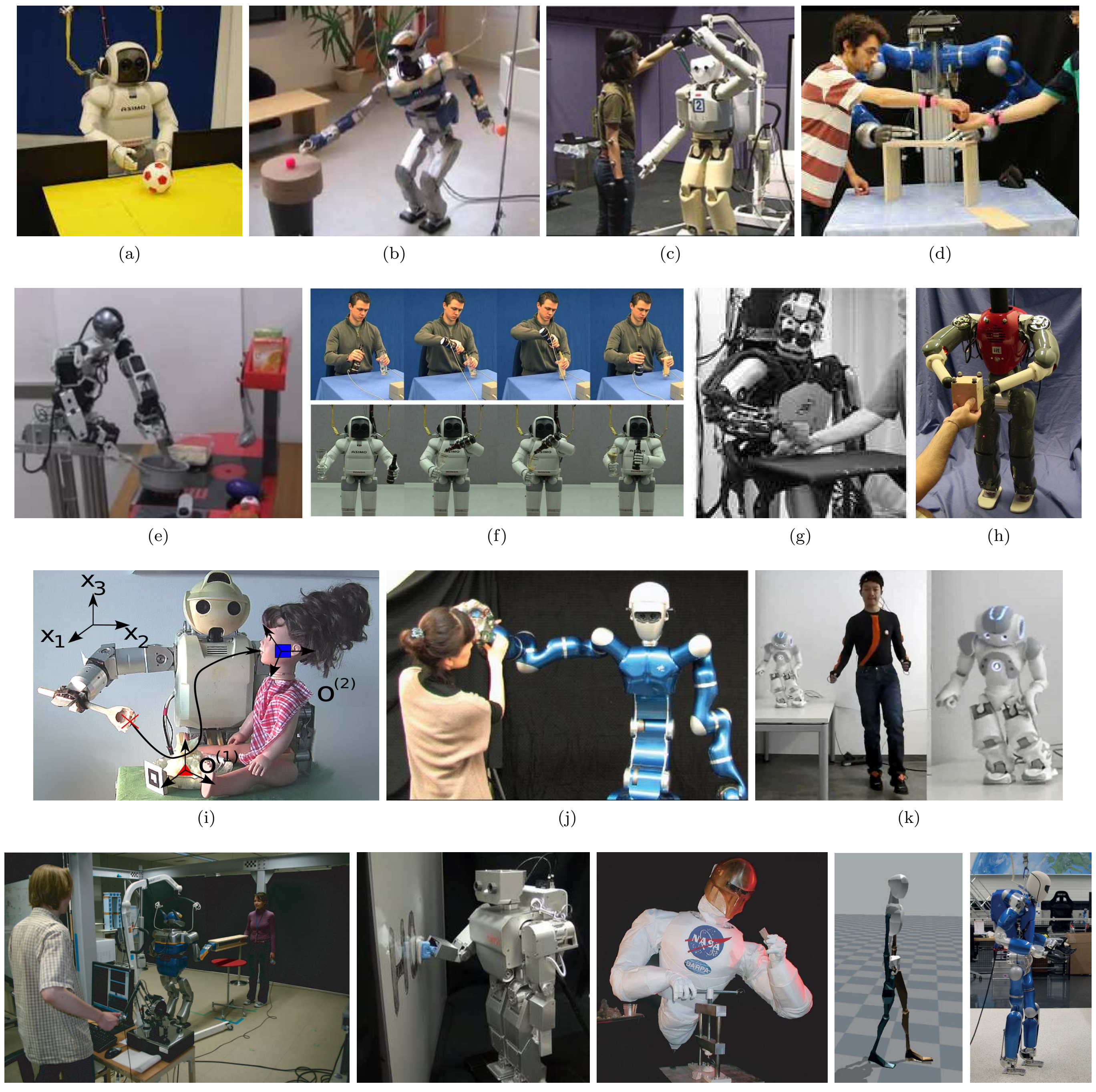


Figure 1: Examples of skills learning in humanoids. (a) Constrained reaching tasks with the Honda ASIMO robot [31]. (b) Task priority recognition with HRP-2 [29]. (c) Communicative gesture synthesis (*high-five* gesture) with the IRT humanoid [56]. (d) Collaborative assembly with a bimanual upper-torso platform composed of two KUKA LWR manipulators [64]. (e) Rice cooking with a humanoid developed at Hanyang University [57]. (f) Bimanual pouring of liquid in a glass with ASIMO [68]. (g) Reaching an object (with hands initially under the table) by using the Sarcos CB-i humanoid [97]. (h) Bimanual adaptation of pointing and reaching with the compliant COMAN humanoid [12]. (i) Feeding task with the Fujitsu HOAP-3 humanoid [11]. (j) Dancing with Justin [55]. (k) Locomotion control acquisition with NAO [32]. (l) Collaborative lifting with HRP-2 [21]. (m) Erasing a whiteboard with the Fujitsu HOAP-2 humanoid [48]. (n) Teleaction with Robonaut [75]. (o) Knee stretched walking pattern acquisition with DLR TORO [33].

observation learning (i.e. whole-body motion retargeting from a human demonstrator to a robot) and is followed by kinesthetic motion refinement. A nonlinear impedance control is designed to achieve the desired behavior during physical coaching: good tracking in case of no physical contact and compliant behavior for physical guidance, with a limited range of compliance. The impedance controller allows both tracking of motion primitives in free-space and a comfortable kinesthetic modification by a human supervisor.

In a user study on kinesthetic teaching [103], the role of kinematic redundancy is identified as one of the main difficulties for the user. An algorithm to teach both end-effector and null-space movements was developed in [83] by utilizing the concept of prioritized tasks. Multiple tasks, such as end-effector and null-space motion primitives, can be taught physically one by one using the compliance controller. In [2], the taught tasks and the user's physical guidance are treated as prioritized tasks and their priorities are adjusted dynamically according to the exerted human force by task transition control.

In contrast to the upper body, kinesthetic teaching for legged humanoid robots has not been studied extensively (namely, where the human's exerted forces influence on the robot's balancing). In [48], an interaction control approach for the upper-body motion was combined with a lower-body balancing algorithm based on the reaction nullspace approach. External forces from kinesthetic teaching or from the task execution were considered as disturbances for the balancer, which employed ankle and hip strategy for balancing. In order to avoid big momentum on upper body to keep ZMP stable, disturbance estimation was used in [72] to take external force into account for explicit balancing and for triggering a compliant behavior at the interaction point. The method helps the user to kinesthetically teach the full-body humanoid robot in a synchronized and easy way, without worrying about balancing.

Besides the aforementioned learning modalities, other strategies have been further utilized for teaching a humanoid robot, which are covered in other dedicated chapters. In particular, teleoperation systems enable the human subject to feel the interaction of a robot with the environment, which can be exploited to learn force profiles for a desired task [21]. Thanks to the advances of artificial skins, tactile contact is used as a modality for learning control by defining specific tactile patterns as control commands. In addition, voice commands and oral feedback can be integrated in the loop of learning control.

3 Movement primitive representations

Movement primitive representations in learning control have the role of robustly and compactly encoding skills, as building blocks that can be organized in parallel and in series to create more complex behaviors. In contrast to the use of movement primitives in the context of motion analysis, movement primitives in the context of learning controllers also require the capability to regenerate movements. The challenge can often be recast as a regression problem; see, e.g., [89] for a review of regression techniques. We will present next several examples of techniques for movement primitives encoding and retrieval.

Many relevant bridges can be built between these different techniques. For this reason, a common notation and terminology is adopted, which can sometimes partly depart from the original work, with the aim of bringing a joint overview of the representations available for learning control. From a structural perspective, these techniques often differ (sometimes in a parametric way) in regard to the spread of the regions in which each model component is valid, from very local behaviors with simple policies changing frequently to global behaviors with complex policies changing only sporadically.

3.1 Weighted least squares

In many humanoid robotics applications, least squares or linear regression appears in one form or another, from simple to large-scale problems. This is mainly due to the fact that an efficient and practical way to handle a nonlinear regression problem is to solve it locally as a linear problem. We first recall here linear regression and its weighted version, which will be at the core of the techniques presented further.

As inputs and outputs are most often multidimensional in humanoid robots applications, we will employ a description of least squares with multidimensional input data organized as $\boldsymbol{X}^{\mathcal{I}} \in \mathbb{R}^{N \times D^{\mathcal{I}}}$ and multidimensional output data organized as $\boldsymbol{X}^{\mathcal{O}} \in \mathbb{R}^{N \times D^{\mathcal{O}}}$, with N the number of datapoints, $D^{\mathcal{I}}$ the dimension of the input and $D^{\mathcal{O}}$ the dimension of the output. The datapoints typically consist of multiple recordings/demonstrations. The concatenation of M recordings of T_m datapoints provides $N = \sum_{m=1}^{M} T_m$ datapoints.

Linear regression aims at finding $\boldsymbol{A} \in \mathbb{R}^{D^{\mathcal{I}} \times D^{\mathcal{O}}}$ such that $\boldsymbol{X}^{\mathcal{O}} = \boldsymbol{X}^{\mathcal{I}} \boldsymbol{A}$. A solution can be found by minimizing the L_2-norm of the residuals

$$\begin{aligned}\hat{\boldsymbol{A}} &= \arg\min_{\boldsymbol{A}} \|\boldsymbol{X}^{\mathcal{O}} - \boldsymbol{X}^{\mathcal{I}}\boldsymbol{A}\|_2 \\ &= \arg\min_{\boldsymbol{A}} (\boldsymbol{X}^{\mathcal{O}} - \boldsymbol{X}^{\mathcal{I}}\boldsymbol{A})^\top (\boldsymbol{X}^{\mathcal{O}} - \boldsymbol{X}^{\mathcal{I}}\boldsymbol{A}), \qquad (1)\end{aligned}$$

which is solved by differentiating with respect to $\boldsymbol{A}$ and equating to zero, providing

$$\begin{aligned}&\hat{\boldsymbol{A}} = (\boldsymbol{X}^{\mathcal{I}\top}\boldsymbol{X}^{\mathcal{I}})^{-1}\boldsymbol{X}^{\mathcal{I}\top}\,\boldsymbol{X}^{\mathcal{O}}, \\ \text{or}\quad &\hat{\boldsymbol{A}} = \boldsymbol{X}^{\mathcal{I}\top}(\boldsymbol{X}^{\mathcal{I}}\boldsymbol{X}^{\mathcal{I}\top})^{-1}\,\boldsymbol{X}^{\mathcal{O}}, \qquad (2)\end{aligned}$$

with residuals (parameters errors) given by $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{A}} = (\boldsymbol{X}^{\mathcal{I}\top}\boldsymbol{X}^{\mathcal{I}})^{-1}$.

The problem above with $D^{\mathcal{I}} = D^{\mathcal{O}} = 1$ can be illustrated in 2D as the problem of fitting a line to a set of datapoints, but it is important to notice that other functions can be fitted by still keeping the problem linear. Indeed, the function does not need to be linear in the input data, only in the parameters that are determined to give the best fit. For example, when fitting trajectories with $\boldsymbol{X}^{\mathcal{I}}$ representing time information, we can consider the relation $\boldsymbol{X}^{\mathcal{O}} = \boldsymbol{X}^{\mathcal{I}}\boldsymbol{A}$ with different forms of inputs such as

$\boldsymbol{X}^{\mathcal{I}} = [t_1, t_2, \ldots, t_N]^\top$ or $\boldsymbol{X}^{\mathcal{I}} = [t_1^2, t_2^2, \ldots, t_N^2]^\top$ while still keeping the model linear in the $\boldsymbol{A}$ parameter.

A typical example is polynomial fitting by treating $\{t, t^2, ...\}$ as distinct independent variables in a multiple regression model, with input data populated with $D^{\mathcal{I}}-1$ derivatives (including $t^0=1$ to learn offset). For polynomials of degree 3 ($D^{\mathcal{I}}=4$), we have

$$\boldsymbol{X}^{\mathcal{I}} = \begin{bmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 & t_N^3 \end{bmatrix}. \tag{3}$$

Polynomial regression is an example of regression using polynomial basis functions to model a relationship between two quantities. The drawback of polynomial fits is that the functions are nonlocal. It is for this reason that the polynomial basis functions are often used in learning control along with other forms of basis functions, such as splines, wavelets, or radial basis functions (RBFs). This last form of RBFs is very popular in robotics, because they are simple and allow the decomposition of a robot behavior into piecewise behavior primitives that are smoothly connected with each other. To allow such combination, the problem above is first reformulated as weighted least squares, by solving the objective

$$\hat{\boldsymbol{A}} = \arg\min_{\boldsymbol{A}} \, (\boldsymbol{X}^{\mathcal{O}} - \boldsymbol{X}^{\mathcal{I}}\boldsymbol{A})^\top \boldsymbol{W} (\boldsymbol{X}^{\mathcal{O}} - \boldsymbol{X}^{\mathcal{I}}\boldsymbol{A}), \tag{4}$$

$$\text{providing} \quad \hat{\boldsymbol{A}} = (\boldsymbol{X}^{\mathcal{I}\top}\boldsymbol{W}\boldsymbol{X}^{\mathcal{I}})^{-1}\boldsymbol{X}^{\mathcal{I}\top}\boldsymbol{W}\,\boldsymbol{X}^{\mathcal{O}}, \tag{5}$$

$$\text{or} \quad \hat{\boldsymbol{A}} = \boldsymbol{W}\boldsymbol{X}^{\mathcal{I}\top}(\boldsymbol{X}^{\mathcal{I}}\boldsymbol{W}\boldsymbol{X}^{\mathcal{I}\top})^{-1}\,\boldsymbol{X}^{\mathcal{O}}, \tag{6}$$

with a weighting matrix $\boldsymbol{W} \in \mathbb{R}^{N\times N}$, and residuals (parameters errors) given by $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{A}} = (\boldsymbol{X}^{\mathcal{I}\top}\boldsymbol{W}\boldsymbol{X}^{\mathcal{I}})^{-1}$. The above problem is ubiquitous for many humanoid robot learning and control models.

3.2 Locally weighted regression (LWR)

Locally weighted regression (LWR) is a direct extension of the weighted least squares formulation in which K weighted regressions are performed on the same dataset $\{\boldsymbol{X}^{\mathcal{I}}, \boldsymbol{X}^{\mathcal{O}}\}$. It aims at splitting a nonlinear problem so that it can be solved locally by linear regression. LWR was introduced by [16] and popularized by [4] in learning control.

LWR computes K estimates $\hat{\boldsymbol{A}}_k$, each with a different weighting function $\phi_k(\boldsymbol{x}_n^{\mathcal{I}})$, often defined as the *radial basis functions* (RBF)

$$\tilde{\phi}_k(\boldsymbol{x}_n^{\mathcal{I}}) = \exp\Big(-\frac{1}{2}(\boldsymbol{x}_n^{\mathcal{I}} - \boldsymbol{\mu}_k^{\mathcal{I}})^\top \boldsymbol{\Sigma}_k^{\mathcal{I}\,-1}(\boldsymbol{x}_n^{\mathcal{I}} - \boldsymbol{\mu}_k^{\mathcal{I}})\Big), \tag{7}$$

or in its rescaled form as (we will see later that the rescaled form is required for some techniques, but for locally weighted regression, it can be omitted to enforce the independence of the local function approximators)

$$\phi_k(\boldsymbol{x}_n^{\mathcal{I}}) = \frac{\tilde{\phi}_k(\boldsymbol{x}_n^{\mathcal{I}})}{\sum_{i=1}^K \tilde{\phi}_i(\boldsymbol{x}_n^{\mathcal{I}})}, \tag{8}$$

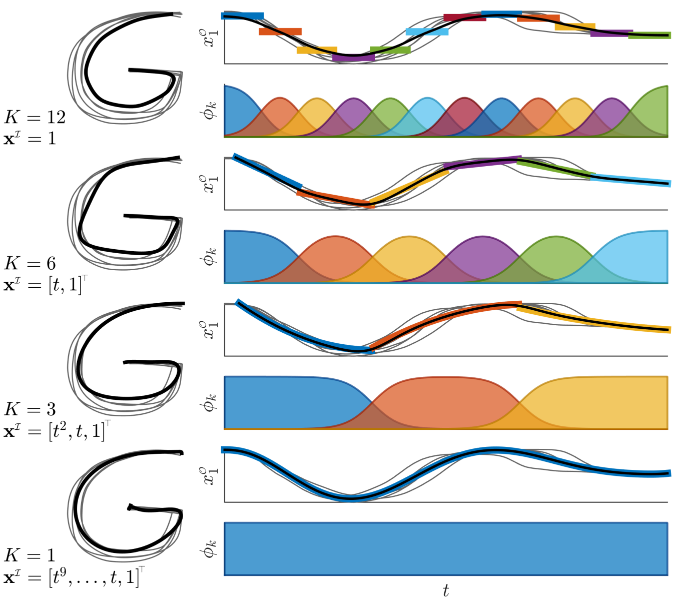


Figure 2: Polynomial fitting with LWR, by considering different degrees of the polynomial and by adopting the number of basis functions accordingly. On one extreme, the top row depicts a local encoding of movement with simple patterns, consequently requiring many basis functions. The bottom row depicts the other extreme with a global polynomial fitting of the same movement requiring polynomials of high degree.

where $\boldsymbol{\mu}_k^{\mathcal{I}}$ and $\boldsymbol{\Sigma}_k^{\mathcal{I}}$ are the parameters of the k-th RBF. An associated diagonal matrix

$$\boldsymbol{W}_k = \text{diag}\Big(\phi_k(\boldsymbol{x}_1^{\mathcal{I}}), \phi_k(\boldsymbol{x}_2^{\mathcal{I}}), \ldots, \phi_k(\boldsymbol{x}_N^{\mathcal{I}})\Big), \tag{9}$$

can then be used with (5) or (6) to evaluate $\hat{\boldsymbol{A}}_k$. The result can finally be used to compute

$$\boldsymbol{X}^{\mathcal{O}} = \sum_{k=1}^{K} \boldsymbol{W}_k \boldsymbol{X}^{\mathcal{I}} \hat{\boldsymbol{A}}_k. \tag{10}$$

Often, the centroids $\boldsymbol{\mu}_k^{\mathcal{I}}$ in (7) are set to uniformly cover the input space, and $\boldsymbol{\Sigma}_k^{\mathcal{I}} = \boldsymbol{I}\sigma^2$ is used as a common bandwidth shared by all basis functions.

LWR can be directly extended to local least squares polynomial fitting by changing the definition of the inputs. Figure 2 shows examples of LWR with various choices of basis functions and numbers of components. Multiple variants of the above formulation exist, including online estimation with a recursive formulation [84], Bayesian treatments of LWR [95], or extensions such as *locally weighted projection regression* (LWPR) that exploit partial least squares to cope with redundant or irrelevant inputs, with an online algorithm to estimate the model parameters incrementally without having to keep the data in memory [98].

Applications in humanoids are diverse, ranging from whole-body inverse dynamics modeling [98] to skillful bimanual control such as devil-stick juggling [5].

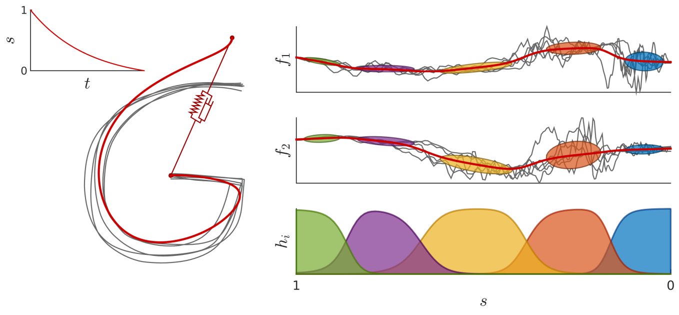


Figure 3: DMP with a GMM to encode the joint distribution of forcing terms $\boldsymbol{f}(s)$ and phase variable s.

3.3 Dynamical movement primitives (DMP)

Dynamical movement primitives (DMP) are popular representations in robotics for learning control. Originally presented in [37], the model has evolved through years with different variants and notations; see [36] for a review. We will use here a notation facilitating the links with the other techniques presented in the chapter. The original DMP uses a unidimensional first-order notation and a neural dynamics formulation of the attractor. At the core of DMP lies a controller in acceleration modulating a spring-damper system with nonlinear forcing terms, defined in its most minimal form by

$$\ddot{\boldsymbol{x}} = k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}) - k^{\mathcal{V}}\dot{\boldsymbol{x}} \;+\; \boldsymbol{f}(s), \qquad \text{with} \quad \boldsymbol{f}(s) = \sum_{k=1}^{K} \phi_k(s)\, s\, \boldsymbol{F}_k. \tag{11}$$

The acceleration command is composed of an attractor to an end-point $\boldsymbol{\mu}_T$ with a spring-damper system of stiffness $k^{\mathcal{P}}$ and damping $k^{\mathcal{V}}$. $\boldsymbol{f}(s)$ represents forcing terms, where s is a phase variable encoding the time evolution of the system. s can be defined in its simplest form as a dynamical system starting from $s=1$ and driven by $\dot{s} = -\alpha s$ to converge to 0 with a given decay factor α; see inset of Fig. 3.

At the beginning of the movement, the nonlinear forcing terms are prevalent and determine the shape of the movement. They then progressively disappear and let the spring-damper system drive entirely the behavior of the system to converge to the attractor point $\boldsymbol{\mu}_T$. An option to handle different motion amplitudes consists of redefining the movement variables as $\boldsymbol{x} \leftarrow \frac{\boldsymbol{x}-\boldsymbol{x}_0}{\boldsymbol{\mu}_T-\boldsymbol{x}_0}$, or multiplying (11) by a scaling factor $(\boldsymbol{\mu}_T - \boldsymbol{x}_0)$; see [36] for details.

By employing the LWR notation from Section 3.2, DMP represents the forcing terms with

$$\boldsymbol{X}^{\mathcal{O}} = \begin{bmatrix} \ddot{\boldsymbol{x}}_1 - k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}_1) + k^{\mathcal{V}}\dot{\boldsymbol{x}}_1 \\ \ddot{\boldsymbol{x}}_2 - k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}_2) + k^{\mathcal{V}}\dot{\boldsymbol{x}}_2 \\ \vdots \\ \ddot{\boldsymbol{x}}_T - k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}_T) + k^{\mathcal{V}}\dot{\boldsymbol{x}}_T \end{bmatrix}, \quad \boldsymbol{X}^{\mathcal{I}} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix},$$
$$\boldsymbol{A}_k = \boldsymbol{F}_k, \quad \boldsymbol{W}_k = \operatorname{diag}\Big(\phi_k(s_1), \phi_k(s_2), \ldots, \phi_k(s_T)\Big). \tag{12}$$

In standard DMP, the centers $\mu_k^{\mathcal{I}}$ of the RBFs defining $\phi_k(s)$ are set at regular time interval, and a variance $\Sigma_k^{\mathcal{I}}$ (constant variance in time) is selected to have a sufficient overlap to guarantee that the forcing terms have smooth profiles. The organization of the receptive fields can alternatively be learned, by either considering the learning of each receptive field separately [36] or globally [14]. As illustrated in Fig. 2, other options to represent the forcing terms are possible. The first column of Fig. 4 shows an example of movement learned by DMP.

Applications in humanoids include the adaptive control of both discrete (point-to-point) and periodic (rhythmic) motions, with experiments such as locomotion [69], reaching while avoiding obstacles [36], interactive rehabilitation exercises in stroke patients [37], playing the drums [97], or cleaning a whiteboard [48].

3.4 Gaussian mixture regression (GMR)

Gaussian mixture regression (GMR) is another popular technique for movement representation, which can be used alone or in conjunction with DMP [12, 14]. It relies on linear transformation and conditioning properties of multivariate normal distributions. GMR provides a synthesis mechanism to compute output distributions in an online manner, with a computation time independent of the number of datapoints used to train the model. A characteristic of GMR is that it does not model the regression function directly. Instead, it first models the joint probability density of the data in the form of a *Gaussian mixture model* (GMM), which can, for example, be estimated by an expectation-maximization (EM) procedure. It can then compute with very low computation the regression function from the learned joint density model.

In GMR, both input and output variables can be multidimensional. Any subset of input-output dimensions can be selected, which can change, if required, at each iteration during reproduction. In humanoids, this can be exploited to handle different sources of missing data, since during reproduction, any combination of input-output mappings can be considered, where expectations on the remaining dimensions can be computed as a multivariate distribution.

In the following, we will denote the block decomposition of a datapoint $\boldsymbol{x}_t$ at time step t, and the center $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ of the i-th Gaussian in the GMM as

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{x}_t^{\mathcal{I}} \\ \boldsymbol{x}_t^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathcal{I}} \\ \boldsymbol{\mu}_i^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{IO}} \\ \boldsymbol{\Sigma}_i^{\mathcal{OI}} & \boldsymbol{\Sigma}_i^{\mathcal{O}} \end{bmatrix}. \tag{13}$$

To make links with the techniques described in the previous sections, we will consider first the example of time-based trajectory retrieval. At each iteration step t during reproduction, $\mathcal{P}(\boldsymbol{x}_t^{\mathcal{O}}|\boldsymbol{x}_t^{\mathcal{I}})$ can be computed as the multimodal conditional distribution

$$\mathcal{P}(\boldsymbol{x}_t^{\mathcal{O}}|\boldsymbol{x}_t^{\mathcal{I}}) \;=\; \sum_{i=1}^{K} h_i(\boldsymbol{x}_t^{\mathcal{I}})\, \mathcal{N}\Big(\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{x}_t^{\mathcal{I}}), \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big) \tag{14}$$

$$\text{with} \quad \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{x}_t^{\mathcal{I}}) \;=\; \boldsymbol{\mu}_i^{\mathcal{O}} + \boldsymbol{\Sigma}_i^{\mathcal{OI}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} (\boldsymbol{x}_t^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}}), \tag{15}$$

$$\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} = \boldsymbol{\Sigma}_i^{\mathcal{O}} - \boldsymbol{\Sigma}_i^{\mathcal{OI}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} \boldsymbol{\Sigma}_i^{\mathcal{IO}} \tag{16}$$

$$\text{and} \quad h_i(\boldsymbol{x}_t^{\mathcal{I}}) \;=\; \frac{\pi_i\, \mathcal{N}(\boldsymbol{x}_t^{\mathcal{I}}|\, \boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}})}{\sum_k^K \pi_k\, \mathcal{N}(\boldsymbol{x}_t^{\mathcal{I}}|\, \boldsymbol{\mu}_k^{\mathcal{I}}, \boldsymbol{\Sigma}_k^{\mathcal{I}})}, \tag{17}$$

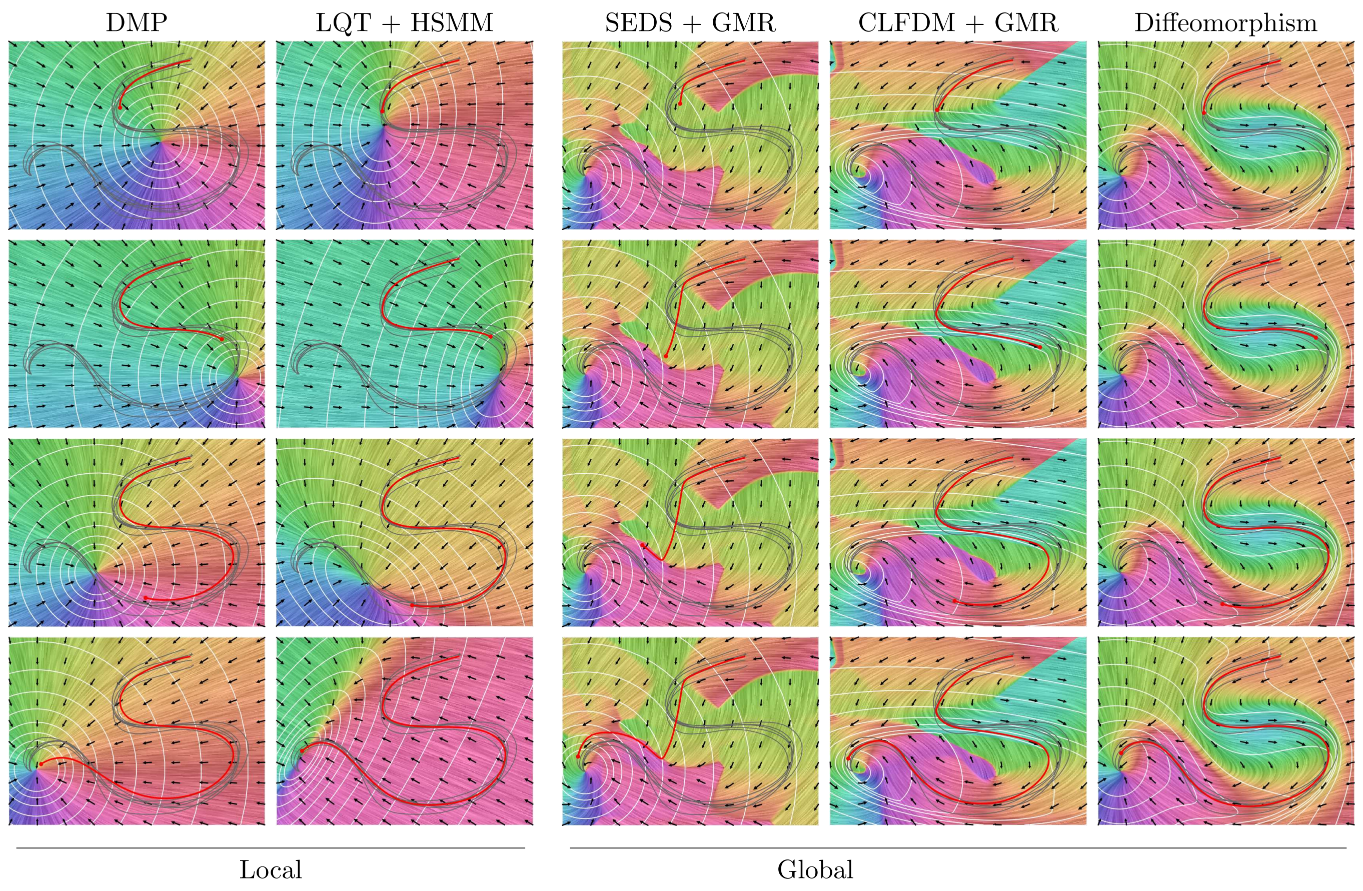


Figure 4: Learning and reproduction of a movement with different forms of motion encoding. The evolution of the flow field for the reproduced motion in red lines is depicted at four different time steps (the four rows). The colored map and black arrows indicate the direction of the flow field. The white contours indicate energy level sets. The first two columns show approaches in which the flow field is local and evolves during the movement (either with an explicit sequence of basis functions as in DMP or with a transition and local duration model as in HSMM). The last three columns present approaches learning a global flow field representing the entire movement. The dataset consists of the first five samples of the *Snake* movement in [40] (in gray lines). **DMP:** With dynamical movement primitives (DMP) [36], the control behavior corresponds to the evolution of an attractor point with an isotropic flow field. $K = 12$ was selected for best results. **LQT+HSMM:** With linear quadratic tracking (LQT), using a stepwise reference retrieved from a hidden semi-Markov model (HSMM) [94, 79], the control behavior corresponds to an adaptive evolution of the attractor point, with a flow field coordinating the variables of the feature space to follow the local trend of the motion (full tracking gain matrices changing over time). This comes at the expense of encoding of full covariances, whose effect is toned down by a lower number of basis functions required in the model (K can be reduced when full covariances are considered). $K = 7$ was selected for best results. **SEDS+GMR:** By using a stable estimator of dynamical system (SEDS) with Gaussian mixture regression (GMR) [40], a full policy can be learned throughout the feature space, which is reflected by a flow field that does not change during the evolution of the motion (autonomous dynamical system). When using an energy function based on the distance to the target, the global stability comes at the expense of constraining the movement to move closer to the target at each iteration (monotonically decreasing distance to the target), which can in some cases distort the movement. $K = 8$ was selected for best results. **CLFDM+GMR:** An extension of SEDS consists of learning energy functions as weighted sums of asymmetric quadratic functions, which are used to redefine the energy levels so that they more closely follow the motion flow [41]. $K = 5$ was selected for best results. **Diffeomorphism:** Approaches based on geometrical diffeomorphic transformations can also be used in autonomous dynamical systems [70, 74]. The approach proposed in [74] is used here to learn energy functions as smooth diffeomorphic mapping of a canonical dynamical system onto the demonstrated trajectories. In this example, the movement is not encoded in a model.

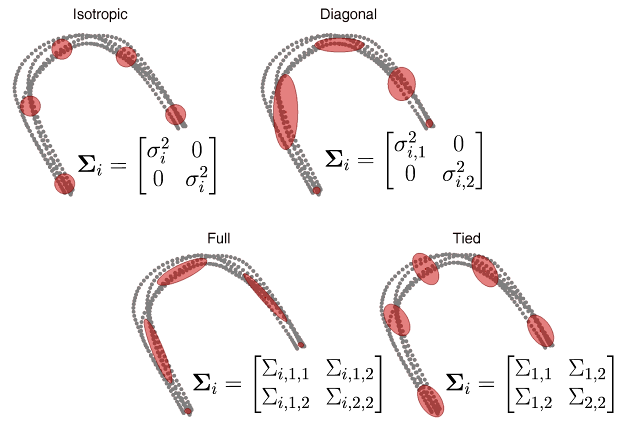

Isotropic
$\Sigma_i = \begin{bmatrix}\sigma_i^2 & 0\\ 0 & \sigma_i^2\end{bmatrix}$
Diagonal
$\Sigma_i = \begin{bmatrix}\sigma_{i,1}^2 & 0\\ 0 & \sigma_{i,2}^2\end{bmatrix}$
Full
$\Sigma_i = \begin{bmatrix}\Sigma_{i,1,1} & \Sigma_{i,1,2}\\ \Sigma_{i,1,2} & \Sigma_{i,2,2}\end{bmatrix}$
Tied
$\Sigma_i = \begin{bmatrix}\Sigma_{1,1} & \Sigma_{1,2}\\ \Sigma_{1,2} & \Sigma_{2,2}\end{bmatrix}$

Figure 5: Different options for the structure of the covariance matrices in Gaussian mixture models.

computed with

$$\mathcal{N}(\boldsymbol{x}_t^{\mathcal{I}}|\,\boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}_i^{\mathcal{I}}|^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}_t^{\mathcal{I}}-\boldsymbol{\mu}_i^{\mathcal{I}})^{\top} \boldsymbol{\Sigma}_i^{\mathcal{I}\,-1}(\boldsymbol{x}_t^{\mathcal{I}}-\boldsymbol{\mu}_i^{\mathcal{I}})\Big). \quad (18)$$

When a unimodal output distribution is required, they resort to the law of total mean and variance to approximate the distribution with the Gaussian

$$\mathcal{P}(\boldsymbol{x}_t^{\mathcal{O}}|\boldsymbol{x}_t^{\mathcal{I}}) = \mathcal{N}\Big(\boldsymbol{x}_t^{\mathcal{O}}|\,\hat{\boldsymbol{\mu}}_t^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}}\Big), \quad \text{with} \quad (19)$$
$$\hat{\boldsymbol{\mu}}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\boldsymbol{x}_t^{\mathcal{I}})\; \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{x}_t^{\mathcal{I}}), \quad \text{and}$$
$$\hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\boldsymbol{x}_t^{\mathcal{I}}) \Big(\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} + \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{x}_t^{\mathcal{I}})\; \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{x}_t^{\mathcal{I}})^{\top}\Big) - \hat{\boldsymbol{\mu}}_t^{\mathcal{O}} \hat{\boldsymbol{\mu}}_t^{\mathcal{O}\top}.$$

Figure 5 shows several types of covariance constraints that are typically used in GMM. Figure 6 illustrates the GMR process with 1D input and 1D output. Figure 3 presents an example of DMP computed through GMR; see also [12, 14]. With the GMR representation, a standard DMP corresponds to a GMM with diagonal covariances. The formulation of DMP as GMR extends DMP so that (1) it allows the encoding of local correlations between the motion variables by extending the diagonal covariances to full covariances; (2) it provides a principled approach to estimate the parameters of the RBFs, similar to a GMM fitting problem; (3) it allows a significant reduction of the number of required RBFs, because the position and spread of each RBF can be automatically adjusted from the demonstrations; and (4) the online estimation of the DMP parameters and the model selection problem (automatically estimating the number of RBFs) can readily exploit techniques compatible with GMM (Bayesian non-parametrics with Dirichlet processes, spectral clustering, small variance asymptotics, etc.).

GMR has been applied in humanoids to learn various tasks, including collaborative transport of objects [21], pouring beverages in a glass [68], tactile correction of humanoid upper-body gestures [82], cooking rice [57], or rolling out pizza dough [13].

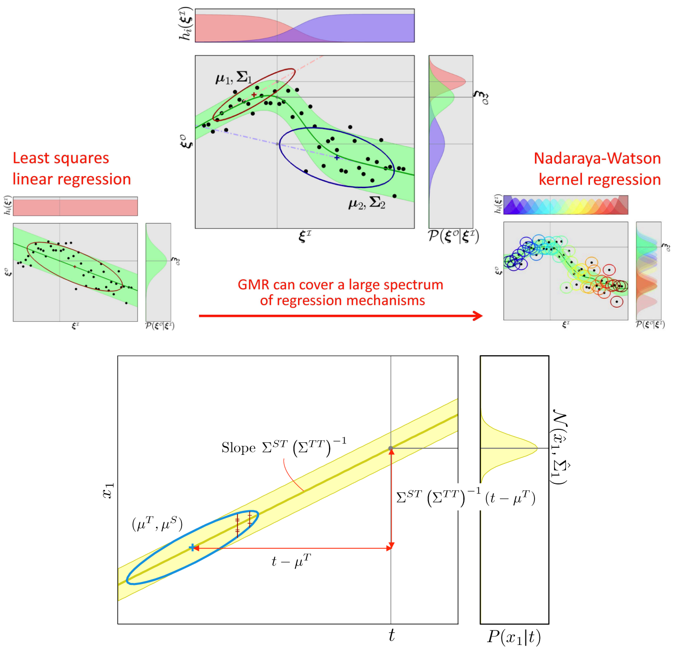


Figure 6: Gaussian mixture regression (GMR) has a simple formulation that can cover a wide range of regression techniques, from multiple multivariate linear regression (single Gaussian) to data-driven kernel-based regression (Gaussian centered on each datapoint).

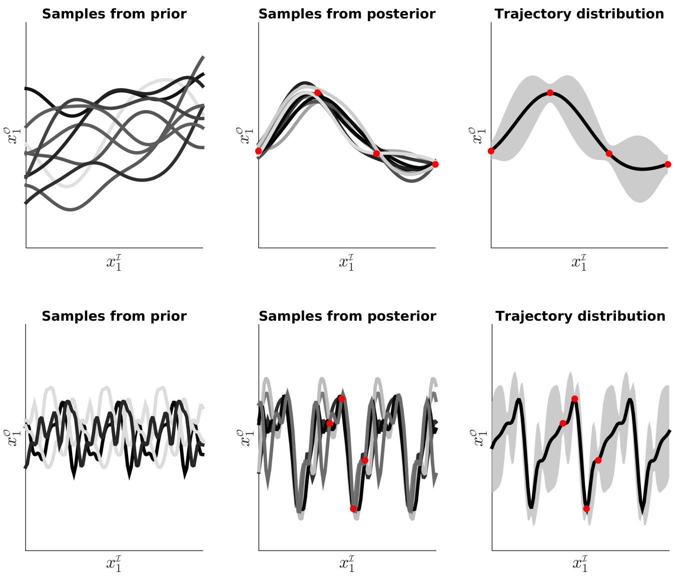


Figure 7: Illustration of Gaussian process regression (GPR) for two different kernels (*top:* RBF, *bottom:* periodic). The first column shows stochastic samples generated from the prior distribution $\boldsymbol{x}^{\mathcal{O}*} \sim \mathcal{N}\big(\boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}*}), \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}*}, \boldsymbol{x}^{\mathcal{I}*})\big)$. The second column shows stochastic samples generated from the posterior distribution $\boldsymbol{x}^{\mathcal{O}*}|\boldsymbol{x}^{\mathcal{O}} \sim \mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ with the datapoints $\{\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{O}}\}$ depicted in red. The last column shows the associated distribution $\mathcal{N}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ with a shaded area corresponding to one standard deviation.

3.5 Gaussian process regression (GPR)

We consider the regression problem of the form $\boldsymbol{x}^{\mathcal{O}} = f(\boldsymbol{x}^{\mathcal{I}}) + \boldsymbol{\eta}$, with f an unknown function and $\boldsymbol{\eta}$ an additive noise process. By assuming the existence of a dataset of observations as input-output pairs $\{\boldsymbol{x}_t^{\mathcal{I}}, \boldsymbol{x}_t^{\mathcal{O}}\}_{t=1}^N$, the goal is to evaluate the form of f and the corresponding output distribution of $\boldsymbol{x}^{\mathcal{O}}$ given previously unseen $\boldsymbol{x}^{\mathcal{I}*}$, namely, $\boldsymbol{x}^{\mathcal{O}*} \sim \mathcal{P}(\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}*})$.

In Section 3.2, we have seen polynomial fitting as an example of parametric modeling technique, where we provided the degree of the polynomial. There are many scenarios in which we have little or no prior knowledge about the appropriate model to use but where we might still have some domain specific knowledge that we would like to express in a more convenient form. For example, we may know that the observations are samples from an underlying process that is smooth, that has typical amplitude, or that the variations in the function take place over known time scales (e.g., within a typical dynamic range). Gaussian processes can be used as a way of reflecting various forms of prior knowledge about the physical process under investigation; see, e.g., [101, 78].

GPR relies on the fact that each observation in a dataset can be imagined as a datapoint sampled from a multivariate Gaussian distribution. The infinite joint distribution over all possible variables is then equivalent to a distribution over a function space. The underlying model still requires hyperparameters to be inferred, but these hyperparameters govern characteristics that are more generic such as the scale of a distribution rather than acting explicitly on the structure or functional form of the signals.

The covariance lies at the core of Gaussian processes, defined through the use of a kernel function $k(\boldsymbol{x}_i^{\mathcal{I}}, \boldsymbol{x}_j^{\mathcal{I}})$ providing the covariance elements between two samples $\boldsymbol{x}_i^{\mathcal{I}}$ and $\boldsymbol{x}_j^{\mathcal{I}}$. For a set of inputs $\boldsymbol{x}^{\mathcal{I}} = \{\boldsymbol{x}_1^{\mathcal{I}}, \boldsymbol{x}_2^{\mathcal{I}}, \ldots, \boldsymbol{x}_N^{\mathcal{I}}\}$, the covariance matrix (also known as the Gram matrix) is then defined as

$$\boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}}) = \begin{bmatrix} k(\boldsymbol{x}_1^{\mathcal{I}}, \boldsymbol{x}_1^{\mathcal{I}}) & k(\boldsymbol{x}_1^{\mathcal{I}}, \boldsymbol{x}_2^{\mathcal{I}}) & \cdots & k(\boldsymbol{x}_1^{\mathcal{I}}, \boldsymbol{x}_N^{\mathcal{I}}) \\ k(\boldsymbol{x}_2^{\mathcal{I}}, \boldsymbol{x}_1^{\mathcal{I}}) & k(\boldsymbol{x}_2^{\mathcal{I}}, \boldsymbol{x}_2^{\mathcal{I}}) & \cdots & k(\boldsymbol{x}_2^{\mathcal{I}}, \boldsymbol{x}_N^{\mathcal{I}}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N^{\mathcal{I}}, \boldsymbol{x}_1^{\mathcal{I}}) & k(\boldsymbol{x}_N^{\mathcal{I}}, \boldsymbol{x}_2^{\mathcal{I}}) & \cdots & k(\boldsymbol{x}_N^{\mathcal{I}}, \boldsymbol{x}_N^{\mathcal{I}}) \end{bmatrix}. \tag{20}$$

This means that the entire function evaluation $f(\boldsymbol{x}^{\mathcal{I}})$ associated with the set of inputs $\boldsymbol{x}^{\mathcal{I}}$ is a sample drawn from a multivariate Gaussian distribution $\boldsymbol{x}^{\mathcal{O}} \sim \mathcal{N}\Big(\boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}}), \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})\Big)$. Therefore a GP specifies a distribution over functions.

We can additionally assume there is noise associated with the observed function values $\boldsymbol{x}_t^{\mathcal{O}}$. Samples are often assumed to be independent and identically distributed (iid), meaning that a term is only added to the diagonal of $\boldsymbol{K}$, giving a modified covariance for noisy observations of the form

$$\tilde{\boldsymbol{K}}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}}) = \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}}) + \Theta^{\text{GP}} \boldsymbol{I}, \tag{21}$$

where $\boldsymbol{I}$ is the identity matrix and Θ^{GP} is a Gaussian process hyperparameter representing the noise variance.

For regression problems, we are interested in the posterior distribution of $\boldsymbol{x}^{\mathcal{O}*}$ given some input datapoint(s) $\boldsymbol{x}^{\mathcal{I}*}$. The **joint distribution** of the demonstrated input-output pair $\boldsymbol{x}^{\mathcal{I}}$ and $\boldsymbol{x}^{\mathcal{O}}$ augmented with $\boldsymbol{x}^{\mathcal{I}*}$ and $\boldsymbol{x}^{\mathcal{O}*}$ is

$$\begin{bmatrix} \boldsymbol{x}^{\mathcal{O}} \\ \boldsymbol{x}^{\mathcal{O}*} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}}) \\ \boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}*}) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}}) & \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}*}) \\ \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}*}, \boldsymbol{x}^{\mathcal{I}}) & \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}*}, \boldsymbol{x}^{\mathcal{I}*}) \end{bmatrix} \right). \tag{22}$$

We can then exploit the conditional probability property of Gaussian distributions (see (14)) to evaluate the posterior distribution over $\boldsymbol{x}^{\mathcal{O}*}$, yielding a Gaussian

$$\boldsymbol{x}^{\mathcal{O}*}|\boldsymbol{x}^{\mathcal{O}} \;\sim\; \mathcal{N}\big(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*\big), \tag{23}$$

with mean and covariance

$$\boldsymbol{\mu}^* = \boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}*}) + \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}*}, \boldsymbol{x}^{\mathcal{I}})\; \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})^{-1} \big(\boldsymbol{x}^{\mathcal{O}} - \boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}})\big),$$
$$\boldsymbol{\Sigma}^* = \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}*}, \boldsymbol{x}^{\mathcal{I}*}) - \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}*}, \boldsymbol{x}^{\mathcal{I}})\; \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})^{-1} \boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}*}).$$

In the above, $\boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})$ can be replaced by $\tilde{\boldsymbol{K}}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})$ if there is noise on the observed function values $\boldsymbol{x}^{\mathcal{O}}$. It is also often assumed in practice that $\begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}}) \\ \boldsymbol{\mu}(\boldsymbol{x}^{\mathcal{I}*}) \end{bmatrix} = \boldsymbol{0}$. Gaussian processes can thus be completely defined by their second-order statistics, where the Gram matrix $\boldsymbol{K}$ is a positive semidefinite covariance built on a scalar product of vectors.

The kernel function is chosen to express a property of similarity so that for points $\boldsymbol{x}_i^{\mathcal{I}}$ and $\boldsymbol{x}_j^{\mathcal{I}}$ that are similar, the corresponding values $\boldsymbol{x}_i^{\mathcal{O}}$ and $\boldsymbol{x}_j^{\mathcal{O}}$ will be more strongly correlated than for dissimilar points. The notion of similarity will depend on the considered humanoid application. Some common aspects that can be defined through the covariance function $k(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})$ are the process stationarity, isotropy, smoothness or periodicity.

When considering continuous time series, it can usually be assumed that past observations can be informative about current data as a function of how long ago they were observed. This can, for example, correspond to a *stationary* covariance dependent on the Euclidean distance $\|\boldsymbol{x}_i^{\mathcal{I}} - \boldsymbol{x}_j^{\mathcal{I}}\|_2$. The process is then considered as *isotropic* and does not depend on directions between $\boldsymbol{x}_i^{\mathcal{I}}$ and $\boldsymbol{x}_j^{\mathcal{I}}$. A process that is both stationary and isotropic is *homogeneous.* In robot learning control, the most employed covariance function of this type is the radial basis function (RBF) that we have seen in Section 3.2. RBF is widely employed when it is expected that nearby inputs $\boldsymbol{x}_i^{\mathcal{I}}$ and $\boldsymbol{x}_j^{\mathcal{I}}$ will have their corresponding outputs $\boldsymbol{x}_i^{\mathcal{O}}$ and $\boldsymbol{x}_j^{\mathcal{O}}$ also nearby (assumption of continuity). When noisy observations $\boldsymbol{x}^{\mathcal{O}}$ are assumed, the kernel can be defined as

$$k(\boldsymbol{x}_i^{\mathcal{I}}, \boldsymbol{x}_j^{\mathcal{I}}) = \Theta_1^{\text{GP}} \exp\!\left(-\frac{1}{\Theta_2^{\text{GP}}} (\boldsymbol{x}_i^{\mathcal{I}} - \boldsymbol{x}_j^{\mathcal{I}})^{\top} (\boldsymbol{x}_i^{\mathcal{I}} - \boldsymbol{x}_j^{\mathcal{I}})\right) + \Theta_3^{\text{GP}} \delta_{i,j}, \tag{24}$$

where $\delta_{i,j} = \mathbb{I}(i = j)$ is equal to one only when $i = j$ and is zero otherwise, resulting in a covariance matrix $\boldsymbol{K}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{I}})$ with noise related to observations only present in the diagonal (noise uncorrelated from sample to sample), scaled by Θ_3^{GP}. The two other hyperparameters Θ_1^{GP} and Θ_2^{GP}, respectively, correspond to output and input scales.

Periodic kernels are another important family of functions for the use of GPR in learning control for humanoids, inducing periodic patterns within the behavior of the process. More complicated covariance functions can also be defined as a linear combination of simpler functions, which

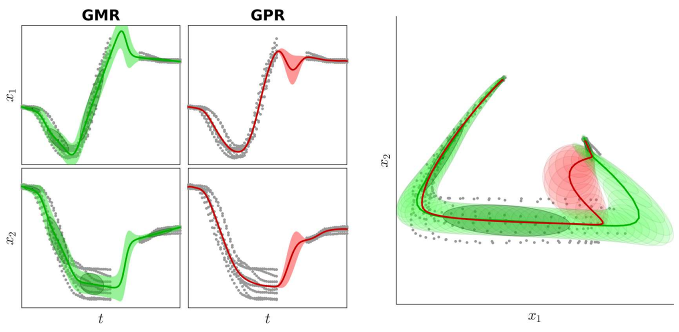


Figure 8: GPR and GMR behaviors for time-based 2D trajectories with a missing portion of the data (see main text for details).

can be exploited to incorporate different insights about the dataset. Another powerful approach to the construction of kernels is to exploit probabilistic models. Given a generative model $\mathcal{P}(\boldsymbol{x}^{\mathcal{I}})$, a valid kernel can be defined as $k(\boldsymbol{x}_i^{\mathcal{I}}, \boldsymbol{x}_j^{\mathcal{I}}) = \mathcal{P}(\boldsymbol{x}_i^{\mathcal{I}})\mathcal{P}(\boldsymbol{x}_j^{\mathcal{I}})$, which can be interpreted as an inner product in the one-dimensional feature space defined by the mapping $\mathcal{P}(\boldsymbol{x}^{\mathcal{I}})$. Namely, two inputs $\boldsymbol{x}_i^{\mathcal{I}}$ and $\boldsymbol{x}_j^{\mathcal{I}}$ will be similar if they both have high probabilities. This approach allows the application of generative models in a discriminative setting, thus combining the respective performance of both generative and discriminative models. This can bring additional properties to the underlying process such as the capability of handling missing data or partial sequences of various lengths (e.g., with HMM as we will see later in the chapter).

Figure 7 shows GPR examples for two different kernels (RBF and periodic). Figure 8 illustrates the differences between GPR and GMR. When using a RBF kernel, GPR retrieves a smooth trajectory, whose missing portion is handled by coming back to the average of the points. The retrieved covariance represents the confidence of the average trajectory estimate, which is here very small and constant for portions of trajectories where datapoints are available, and that grows for missing portions of the data. Each output variable is retrieved individually, thus generating a diagonal covariance in the x_1-x_2 graph. In the case of GMR, the datapoints are first used to learn a joint distribution in the form of a GMM (here, with four Gaussians modeling the joint distribution t-x_1-x_2). The dataset is then discarded. GMR consists of computing a weighted sum of conditional distributions and approximating it with a Gaussian; see Section 3.4. The missing portion of the data is handled by a smooth transition between the two linear trends on the two sides of the missing data. In contrast to GPR whose variances represent the confidence on the mean estimate, the retrieved covariances in GMR represent the variations and correlations observed in the data, with full covariances retrieved for the output distributions, indicating local coordination patterns (green ellipsoids in the x_1-x_2 graph). Similarly to the mean estimate, the missing portion of the data is handled by smoothly interpolating between the observed covariance on the two sides. Figure 8 then shows that depending on the control policy learning problem considered, one or the other approach can be preferred. It should first be noted that GMR is tightly linked to multivariate normal properties (in particular, for Gaussian conditioning), while GPR is not restricted to the use of Gaussian kernels and is thus more generic. GMR is competitive when fast computation is required, when the input and output components can change (e.g., estimation from partial input observation), or when an estimate of the variation and coordination of a multivariate output signal is required. This comes at the expense of requiring the estimation of a GMM, together with the number of Gaussians used in the model. When coordination and variation information is important, a generalized Wishart process can alternatively be considered as a substitute for GP to model the evolution of covariances [102].

Various applications of GPR have been proposed for robot learning control. In [22], GPR is exploited in a humanoid tracking and reaching movement, in which a set of external task parameters is associated with DMP parameters encoding movements, and where new task parameters are used to generate movements with GPR in an online manner. In [85], a sparse GP model is developed for the control of a PR2 robot, with an efficient online selection of the training points to learn inverse dynamics models from large datasets. In [99], GPR is applied to a 2D bipedal walking problem, where GPR is used to generalize a suboptimal joint trajectory by using previous optimization results without running expensive nonlinear optimization procedures.

3.6 Trajectory distributions

This section discusses different approaches to represent multiple recordings of movements as trajectory distributions expressed in a compact form. A basic way of representing a collection of M trajectories in a probabilistic form is to reorganize each trajectory as a hyperdimensional datapoint $\boldsymbol{\xi}_m = [\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_T^\top]^\top \in \mathbb{R}^{DT}$, and fitting a Gaussian $\mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\xi}}, \boldsymbol{\Sigma}^{\boldsymbol{\xi}})$ to these datapoints. Since the dimension DT might be much larger than the number of datapoints M, a prior usually needs to be defined for the estimation of the covariance, such as $\mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\xi}}, \boldsymbol{\Sigma}^{\boldsymbol{\xi}} + \beta^{-1}\boldsymbol{I})$ (Tikhonov regularization). An eigendecomposition can also be used to estimate only the first few eigencomponents. Expressed in a matrix form, we have

$$\boldsymbol{\Sigma}^{\boldsymbol{\xi}} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top = \sum_{j=1}^{D} \lambda_j \boldsymbol{v}_j \boldsymbol{v}_j^\top, \tag{25}$$

with $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_D]$ and $\boldsymbol{D} = \operatorname{diag}(\lambda_1^2, \lambda_2^2, \ldots, \lambda_D^2)$, and a regularized covariance with minimal admissible eigenvalue can be computed as $\boldsymbol{\Sigma}^{\boldsymbol{\xi}} \leftarrow \boldsymbol{V}\tilde{\boldsymbol{D}}\boldsymbol{V}^\top$ with $\tilde{\boldsymbol{D}} = \operatorname{diag}(\tilde{\lambda}_1^2, \tilde{\lambda}_2^2, \ldots, \tilde{\lambda}_D^2)$ and $\tilde{\lambda}_j = \max(\tilde{\lambda}_j, \lambda_{\min})\ \forall j \in \{1, \ldots, D\}$.

Representing a collection of trajectories in the form of a multivariate distribution has several advantages. With such representation, new trajectories can be stochastically generated with

$$\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\xi}}, \boldsymbol{\Sigma}^{\boldsymbol{\xi}}) \iff \boldsymbol{\xi} \sim \boldsymbol{\mu}^{\boldsymbol{\xi}} + \boldsymbol{V}\tilde{\boldsymbol{D}}^{\frac{1}{2}}\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{26}$$

and the conditional probability property (see (14)) can be exploited to generate trajectories passing through via-points (including starting and/or ending points). This is simply achieved by specifying as inputs $\boldsymbol{\xi}^{\mathcal{I}}$ in (14) the datapoints the system needs to pass through (with corresponding dimensions in the hyperdimensional vector) and by retrieving as output $\boldsymbol{\xi}^{\mathcal{O}}$ the remaining parts of the trajectory.

Next, we will present two techniques to retrieve trajectory distributions in a more parsimonious manner, by either relying on RBF or GMM encoding.

3.6.1 ProMP

The ProMP (probabilistic movement primitive) model [73] assumes that each demonstrated trajectory $m \in \{1,\ldots,M\}$ can be approximated by a weighted sum of K normalized RBFs with

$$\boldsymbol{\xi}_m = \boldsymbol{\Psi}\boldsymbol{w}_m + \boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \lambda\boldsymbol{I}), \tag{27}$$

and basis functions defined similarly as LWR and DMP, organized as

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{I}\phi_1(t_1) & \boldsymbol{I}\phi_2(t_1) & \cdots & \boldsymbol{I}\phi_K(t_1) \\ \boldsymbol{I}\phi_1(t_2) & \boldsymbol{I}\phi_2(t_2) & \cdots & \boldsymbol{I}\phi_K(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}\phi_1(t_T) & \boldsymbol{I}\phi_2(t_T) & \cdots & \boldsymbol{I}\phi_K(t_T) \end{bmatrix}, \tag{28}$$

with $\boldsymbol{\Psi} \in \mathbb{R}^{DT\times DK}$ and $\boldsymbol{I} \in \mathbb{R}^{D\times D}$. A vector $\boldsymbol{w}_m \in \mathbb{R}^{DK}$ can be estimated for each of the M demonstrated trajectories by the least squares estimate

$$\boldsymbol{w}_m = (\boldsymbol{\Psi}^\top\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^\top\boldsymbol{\xi}_m. \tag{29}$$

By assuming that $\{\boldsymbol{w}_m\}_{m=1}^M$ can be represented with a Gaussian $\mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{w}}, \boldsymbol{\Sigma}^{\boldsymbol{w}})$ characterized by a center $\boldsymbol{\mu}^{\boldsymbol{w}} \in \mathbb{R}^{DK}$ and a covariance $\boldsymbol{\Sigma}^{\boldsymbol{w}} \in \mathbb{R}^{DK\times DK}$, a trajectory distribution $\mathcal{P}(\boldsymbol{\xi})$ can then be retrieved by integrating out $\boldsymbol{w}$,

$$\mathcal{P}(\boldsymbol{\xi}) = \int \mathcal{P}(\boldsymbol{\xi}|\boldsymbol{w})\mathcal{P}(\boldsymbol{w})\, d\boldsymbol{w}, \tag{30}$$

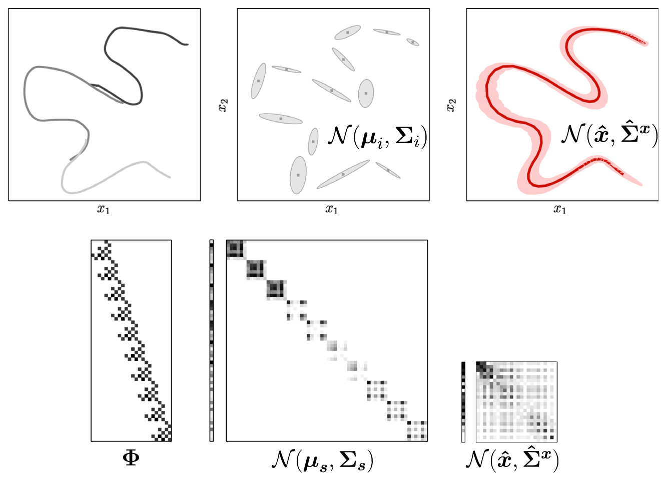
resulting in the trajectory distribution

$$\boldsymbol{\xi} \sim \mathcal{N}\Big(\boldsymbol{\Psi}\boldsymbol{\mu}^{\boldsymbol{w}} \;,\; \boldsymbol{\Psi}\boldsymbol{\Sigma}^{\boldsymbol{w}}\boldsymbol{\Psi}^\top + \lambda\boldsymbol{I}\Big), \tag{31}$$

with $\boldsymbol{\xi} \in \mathbb{R}^{DT}$ a trajectory of T datapoints of D dimensions organized in a vector form and $\boldsymbol{I} \in \mathbb{R}^{DT\times DT}$.

The ProMP parameters are λ, $\mu_k^{\mathcal{I}}$, $\Sigma_k^{\mathcal{I}}$, $\boldsymbol{\mu}^{\boldsymbol{w}}$, and $\boldsymbol{\Sigma}^{\boldsymbol{w}}$. As for DMP, the parameters of the RBFs $\mu_k^{\mathcal{I}}$ and $\Sigma_k^{\mathcal{I}}$ are usually predefined by the experimenter, with centers $\mu_k^{\mathcal{I}}$ equally spread in time and a constant variance $\Sigma_k^{\mathcal{I}} = \sigma^2$ set to provide a sufficient overlap of the basis functions. A Gaussian of DK dimensions is estimated (instead of the DT dimensions in (26)), providing a compact representation of the movement, separating the temporal components $\boldsymbol{\Psi}$ and spatial components $\mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{w}}, \boldsymbol{\Sigma}^{\boldsymbol{w}})$. Similarly to DMP, ProMP can be coupled with GMM/GMR to automatically estimate the positioning and spread of the basis functions as a joint distribution problem, instead of specifying them manually.

ProMP has been demonstrated in varied tasks requiring humanlike motion capabilities such as table tennis strokes [80], playing the maracas, or handling a hockey stick [73], as well as for collaborative object handover and assistance in box assembly [64].



x_2
x_1
$\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$
$\mathcal{N}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}})$
$\boldsymbol{\Phi}$
$\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{s}})$

Figure 9: GMM with dynamic features to construct a trajectory distribution. The top row depicts a 2D example, showing that the approach does not require time alignment and can readily exploit piecewise demonstrations of a movement. The bottom row illustrates, with a short trajectory, the involved matrices depicted as blocks with levels of gray proportional to the absolute values of the matrix elements (white color depicts zero entries). This shows the sparsity of the different operators and the resulting full covariance representing the trajectory distribution.

3.6.2 Trajectory-GMM

The exploitation of statistics from both static and dynamic features of a GMM, for the purpose of generating data, originates from the field of speech processing [23]. Also called trajectory-GMM, this approach has a long history and is considered as a standard technique in this field, in particular when employed in the context of hidden Markov models (HMM).

In robotics, it provides a simple approach to synthesize trajectories without discontinuities even when a small number of Gaussians are used to encode the movement. This is achieved by coordinating the distributions of both static and dynamic features in the considered time series. For the encoding of movements, velocity and acceleration can be used as dynamic features [9]. By considering $\boldsymbol{x}_t \in \mathbb{R}^D$ as a multivariate position vector, with an estimation of velocity (and higher-order derivatives) from two consecutive time steps, we define an observation vector $\boldsymbol{\zeta}_t \in \mathbb{R}^{DC}$ as the concatenated position, velocity and acceleration ($C=3$) vectors at time step t, namely,[1]

$$\boldsymbol{\zeta}_t = \begin{bmatrix} \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \\ \ddot{\boldsymbol{x}}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} \\ \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \\ \boldsymbol{x}_{t+2} \end{bmatrix}. \tag{32}$$

$\boldsymbol{\zeta} = \big[\boldsymbol{\zeta}_1^\top, \boldsymbol{\zeta}_2^\top, \ldots, \boldsymbol{\zeta}_T^\top\big]^\top$ and $\boldsymbol{x} = \big[\boldsymbol{x}_1^\top, \boldsymbol{x}_2^\top, \ldots, \boldsymbol{x}_T^\top\big]^\top$ are then defined as large vectors concatenating $\boldsymbol{\zeta}_t$ and $\boldsymbol{x}_t$ for all time steps.

Similarly to the matrix operator (32) defined for a single time step, a large sparse matrix $\boldsymbol{\Phi}$ can be defined so that

[1] To simplify the notation, the number of derivatives is set up to acceleration ($C=3$), but the results can easily be generalized to a higher or lower number of derivatives.

$\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{x}$, namely,[2]

$$\overbrace{\begin{bmatrix} \vdots \\ \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \\ \ddot{\boldsymbol{x}}_t \\ \boldsymbol{x}_{t+1} \\ \dot{\boldsymbol{x}}_{t+1} \\ \ddot{\boldsymbol{x}}_{t+1} \\ \vdots \end{bmatrix}}^{\boldsymbol{\zeta}} = \overbrace{\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \iddots & \\ \cdots & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \\ \cdots & -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} & \cdots & \\ \cdots & \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I} & \cdots & \\ & \cdots & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \cdots \\ & \cdots & -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} & \cdots \\ & \cdots & \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I} & \cdots \\ & \iddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}}^{\boldsymbol{\Phi}} \overbrace{\begin{bmatrix} \vdots \\ \boldsymbol{x}_t \\ \boldsymbol{x}_{t+1} \\ \boldsymbol{x}_{t+2} \\ \boldsymbol{x}_{t+3} \\ \vdots \end{bmatrix}}^{\boldsymbol{x}}. \tag{33}$$

A GMM is used to model a dataset $\{\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2, \ldots, \boldsymbol{\zeta}_N\}$. The GMM parameters $\boldsymbol{\Theta}^{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^K$, with $\boldsymbol{\mu}_i \in \mathbb{R}^{DC}$ and $\boldsymbol{\Sigma}_i \in \mathbb{R}^{DC\times DC}$, represent the mixing coefficients, centers, and covariances of the Gaussian components, which can, for example, be estimated by an expectation-maximization (EM) procedure.

During reproduction, a sequence of states $\boldsymbol{s} = \{s_1, s_2, \ldots, s_T\}$ of T time steps is first generated (or retrieved from a demonstration, as we will see later in Section 3.7). With discrete states $s_t \in \{1, \ldots, K\}$, the likelihood of a movement $\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{x}$ is given by

$$\mathcal{P}(\boldsymbol{\zeta}|\boldsymbol{s}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\zeta}_t \,|\, \boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t}), \tag{34}$$

where $\boldsymbol{\mu}_{s_t}$ and $\boldsymbol{\Sigma}_{s_t}$ are the center and covariance of state s_t at time step t. This product can be rewritten as

$$\mathcal{P}(\boldsymbol{\Phi}\boldsymbol{x}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\Phi}\boldsymbol{x} \,|\, \boldsymbol{\mu}_{\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{s}}),$$

with $\boldsymbol{\mu}_{\boldsymbol{s}} = [\boldsymbol{\mu}_{s_1}^\top, \boldsymbol{\mu}_{s_2}^\top, \ldots, \boldsymbol{\mu}_{s_T}^\top]^\top$ and $\boldsymbol{\Sigma}_{\boldsymbol{s}} = \text{blockdiag}(\boldsymbol{\Sigma}_{s_1}, \boldsymbol{\Sigma}_{s_2}, \ldots, \boldsymbol{\Sigma}_{s_T})$.

By using the relation $\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{x}$, a trajectory can then be retrieved by solving

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} \; \log \mathcal{P}(\boldsymbol{\Phi}\boldsymbol{x} \,|\, \boldsymbol{s}), \tag{35}$$

resulting in a trajectory distribution $\boldsymbol{x} \sim \mathcal{N}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}})$ with parameters

$$\hat{\boldsymbol{x}} = \big(\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \boldsymbol{\Phi}\big)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \boldsymbol{\mu}_{\boldsymbol{s}}, \quad \hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}} = \sigma \big(\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \boldsymbol{\Phi}\big)^{-1}, \tag{36}$$

where $\hat{\boldsymbol{x}} \in \mathbb{R}^{DT}$ is the average trajectory stored in a vector form, σ is a scale factor, $\boldsymbol{\Phi} \in \mathbb{R}^{DCT\times DT}$, $\boldsymbol{\Sigma}_{\boldsymbol{s}} \in \mathbb{R}^{DCT\times DCT}$, and $\boldsymbol{\mu}_{\boldsymbol{s}} \in \mathbb{R}^{DCT}$.

Some links with ProMP (see §3.6.1) can be drawn. In trajectory-GMM, the smoothness of the retrieved movements is ensured by the encoding of derivatives, where the number of derivatives will influence the width of the band-diagonal structure of the trajectory covariances. We can indeed notice that for $C = 1$, we have $\boldsymbol{\Phi} = \boldsymbol{I}$ in (33), and (36) collapses to a stepwise trajectory with $\hat{\boldsymbol{x}} = \boldsymbol{\mu}_{\boldsymbol{s}}$ and $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}} = \sigma \boldsymbol{\Sigma}_{\boldsymbol{s}}$. We can further note that similarly to (31), $\boldsymbol{\mu}_{\boldsymbol{s}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{s}}$ can be constructed with $\boldsymbol{\mu}_{\boldsymbol{s}} = \tilde{\boldsymbol{\Psi}} \boldsymbol{\mu}^{\boldsymbol{w}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{s}} = \tilde{\boldsymbol{\Psi}} \tilde{\boldsymbol{\Sigma}}^{\boldsymbol{w}} \tilde{\boldsymbol{\Psi}}^\top$, where $\tilde{\boldsymbol{\Psi}}$ is a binary version of $\boldsymbol{\Psi}$ in (28) and $\tilde{\boldsymbol{\Sigma}}^{\boldsymbol{w}}$ is a block diagonal version of $\boldsymbol{\Sigma}^{\boldsymbol{w}}$ in (31).

[2] Note that a similar operator is defined to handle border conditions and that $\boldsymbol{\Phi}$ can automatically be constructed with the Kronecker product operator.

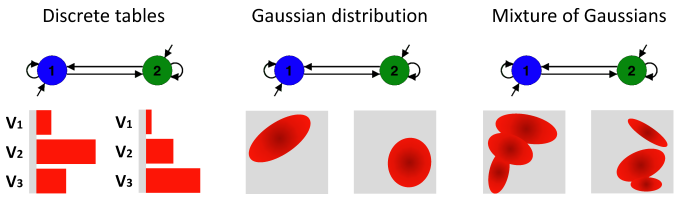


Figure 10: The observation model associated with each component of the HMM can take various forms.

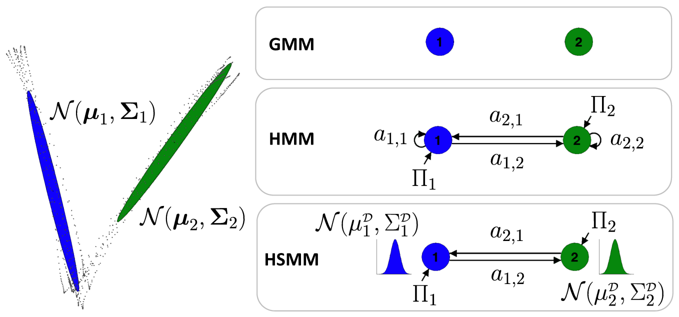

$\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$
$\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$
GMM
HMM
HSMM
$a_{1,1}$
$a_{2,1}$
$a_{1,2}$
$a_{2,2}$
Π_1
Π_2
$\mathcal{N}(\mu_1^{\mathcal{D}}, \Sigma_1^{\mathcal{D}})$
$\mathcal{N}(\mu_2^{\mathcal{D}}, \Sigma_2^{\mathcal{D}})$
1
2

Figure 11: Difference of model structures in GMM, HMM and HSMM encoding.

Figure 9 illustrates the trajectory-GMM approach. Such trajectory distribution encoding with dynamic features has been exploited in robotics for humanlike motion planning and control [92, 9, 79].

3.7 Hidden Markov models

It is often important in learning control to model the transitions between several regimes or between the motion primitives described in the previous sections. As we will see next, a first-order Markov assumption is often adopted to simplify the model acquisition process.

3.7.1 Markov model (MM)

With a *first-order Markov model*, the joint distribution of a sequence of states $\{s_t\}_{t=1}^T$ is assumed to be of the form

$$\mathcal{P}(s_1, s_2, \ldots, s_T) = \mathcal{P}(s_1) \prod_{t=2}^{T} \mathcal{P}(s_t|s_{t-1}), \tag{37}$$

thus providing

$$\mathcal{P}(s_t|s_1, s_2, \ldots, s_{t-1}) = \mathcal{P}(s_t|s_{t-1}). \tag{38}$$

Often, the conditional distributions $\mathcal{P}(s_t|s_{t-1})$ is assumed to be *stationary* (*homogeneous Markov chain*). A *transition matrix* $\boldsymbol{A}$ can then be defined, where

$$a_{i,j} = \mathcal{P}(s_{t+1} = j \,|\, s_t = i) \tag{39}$$

is the probability of getting from state i to state j in one step.

Similarly, an *initial state distribution* can be defined by

$$\Pi_i = \mathcal{P}(s_1 = i) \quad \text{with} \quad \sum_{i=1}^{K} \Pi_i = 1. \tag{40}$$

The parameters of a Markov model of K components are described as

$$\boldsymbol{\Theta}^{\text{MM}} = \big\{ \{a_{i,j}\}_{j=1}^K, \Pi_i \big\}_{i=1}^K. \tag{41}$$

3.7.2 Hidden Markov model (HMM)

HMM is used in many fields for time series or sequence analysis, or in fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are) [77]. This is typically the case in humanoid robots, where the generative aspect of the model can additionally be exploited for synthesis purpose.

An HMM is composed of a set of hidden states whose output can be described in different forms; see Fig. 10. The most employed form in humanoid learning control is to encode each output as a multivariate Gaussian. In the context of robotics, we can then think of an HMM either as a Markov chain with hidden states and stochastic measurements, or as a GMM with latent variables changing over time. The parameters of an HMM of K components, with a single Gaussian as output distribution, are described as

$$\boldsymbol{\Theta}^{\text{HMM}} = \big\{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big\}_{i=1}^{K}. \tag{42}$$

HMMs are most often trained by an expectation-maximization procedure [77], which guarantee to converge to a local optimum from an initial estimate. However, other techniques based on spectral learning or methods of moments exist, aiming at finding global estimates; see, e.g., [3].

HMMs have been exploited in [52] for hand drawing gestures, in [56] for communicative gestures with a humanoid robot (such as producing "high-five" hand gestures or conducting an orchestra), in [51] for online learning of full-body motion primitives, or in [93] for the autonomous acquisition of motion symbols in a humanoid robot.

3.7.3 Hidden semi-Markov model (HSMM)

The state duration in standard HMM is indirectly modeled through self-transition probabilities and follows a geometric distribution

$$\mathcal{P}(d) = a_{i,i}^{d-1}(1 - a_{i,i}), \tag{43}$$

which is not very accurate to model duration information. Instead of relying on self-transition probabilities to estimate the above probability, the *hidden semi-Markov model* (HSMM) provides an explicit model of the state duration [77]. The parameters of an HSMM of K components, where each component $i \in \{1, \ldots, K\}$ is characterized by a single Gaussian $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ as output distribution and a single lognormal distribution $\mathcal{LN}(\mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}})$ as state duration, are described by

$$\boldsymbol{\Theta}^{\text{HSMM}} = \big\{\{a_{i,j}\}_{j=1, j\neq i}^{K}, \Pi_i, \mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\big\}_{i=1}^{K}, \tag{44}$$

which can be trained by an expectation-maximization procedure. HSMM can be jointly used with dynamic features as in Section 3.6.2 to generate sequences of states from the model parameters [9].

From a learning control perspective, the duration model in HSMM can be viewed as a local time duration encoding, with a timer starting once we enter the state. This relative duration encoding allows the encoding of both discrete and periodic movements. Such model can be trained from a set of partial observations of a movement, without requiring temporal alignment. On the two ends of the spectrum, a duration model as a flat distribution corresponds to a time-independent state, and a peaked distribution corresponds to a transition occurring at a specific duration after which we enter the state.

Figure 11 illustrates the differences of structure in hidden Markov models (including standard GMM without transition information). The second column of Fig. 4 shows an illustrative example of movement learned by HSMM, and coupled with a linear quadratic controller.

The HSMM has been exploited for humanlike robot manipulation skills to learn controllers robust to both time and space discrepancies [104, 94, 79].

3.8 Autonomous dynamical systems

Another approach to represent movements and skills in humanoids is to encode the entire attractor landscape in the state-space of the observed data. Such approach can provide representations based on time-invariant autonomous systems. It usually comes at the expense of estimating asymptotically stable dynamical systems in high-dimensional spaces. The GMR representation (presented in the context of trajectories to estimate $\mathcal{P}(\boldsymbol{x}|t)$ in Section 3.4) could, for example, be employed to retrieve an autonomous system $\mathcal{P}(\dot{\boldsymbol{x}}|\boldsymbol{x})$ from the joint distribution $\mathcal{P}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ encoded in a GMM [30, 11], by computing iteratively velocity commands

$$\hat{\dot{\boldsymbol{x}}} = \sum_{i=1}^{K} h_i(\boldsymbol{x}) \Big(\overbrace{\boldsymbol{\Sigma}_i^{\dot{\boldsymbol{x}}\boldsymbol{x}} (\boldsymbol{\Sigma}_i^{\boldsymbol{x}})^{-1}}^{\boldsymbol{A}_i} \boldsymbol{x} + \overbrace{\boldsymbol{\mu}_i^{\dot{\boldsymbol{x}}} - \boldsymbol{\Sigma}_i^{\dot{\boldsymbol{x}}\boldsymbol{x}} (\boldsymbol{\Sigma}_i^{\boldsymbol{x}})^{-1} \boldsymbol{\mu}_i^{\boldsymbol{x}}}^{\boldsymbol{b}_i} \Big). \tag{45}$$

The *stable estimator of dynamical systems* (SEDS) [40] relies on this regression strategy, by replacing the standard EM procedure commonly used to estimate the GMM parameters with a constrained optimization procedure taking into account not only log-likelihood optimization but also constraining the parameters to form a contractive system [26]. This can be achieved by constraining the poles of the dynamical system to have strictly negative real parts, corresponding to $\boldsymbol{A}_i + \boldsymbol{A}_i^{\top}$ being negative definite. All components in the GMM must also define the same attractor point $\boldsymbol{x}_T$, corresponding to the constraint $\boldsymbol{b}_i = -\boldsymbol{A}_i \boldsymbol{x}_T$. Such constraints provide global asymptotic stability guarantee to the estimated GMM parameters when using (45).

SEDS results in fast and reactive movements characterized by a decaying distance to the target, which can distort the original paths in some cases. This problem is tackled in [41] by associating a different control Lyapunov function (energy function) to the autonomous system. This energy function is parametrized as a weighted sum of asymmetric quadratic functions, guaranteeing to asymptotically reach the target point. The resulting *control Lyapunov function-based dynamic movements* (CLFDM) approach proceeds in three steps: (1) learning a Lyapunov function from the demonstrations by solving a constrained optimization problem; (2) using a regression technique to model an (unstable) estimate of the motion from the demonstrations; and (3) using (1) to ensure stability of (2) during the task execution by solving at runtime a constrained convex optimization problem for online correction. An example with GMR used as regression technique is shown in the third column of Fig. 4.

Approaches based on geometrical diffeomorphic transformation have also been investigated [70, 74]. In [74], the objective is to learn a smooth diffeomorphism that maps line segments onto the demonstrated trajectories, where the line segments are characterized by the start and the end of the movement (at the origin), corresponding to orbits of the dynamical system $\dot{\boldsymbol{x}} = -\boldsymbol{x}$). The smoothness of the mapping is achieved by a composition of locally weighted translations from a set of RBFs. The approach allows the parametrization of the flow field to obtain different correction/tracking behaviors when moving/starting away from the demonstrated trajectories (by modulating how strongly the system comes back to the reference trajectories). An example is shown in the last column of Fig. 4.

The GMR, SEDS, CLFDM, and diffeomorphic approaches in the above have been demonstrated with humanoids in experiments including packing tasks [30], feeding tasks [11], reproducing natural upper-body gestures [40], contouring obstacles [70], or catching objects in flight [44].

4 Learning controllers

While Section 3 discussed movement primitives encoding and retrieval, we discuss here how such representation can be more tightly integrated with existing tracking and regulation control strategies employed in humanoids.

4.1 Linear quadratic tracking (LQT) and regulation (LQR)

The previous section discussed the problem of generating and adapting reference trajectories, by assuming that a controller is available to track the retrieved reference. In this section, the problem is extended to that of directly estimating a controller $\boldsymbol{u}_t$ for a discrete linear dynamical system

$$\boldsymbol{\zeta}_{t+1} = \boldsymbol{A}\boldsymbol{\zeta}_t + \boldsymbol{B}\boldsymbol{u}_t, \tag{46}$$

with state variable $\boldsymbol{\zeta}_t = \big[\boldsymbol{x}_t^\top, \dot{\boldsymbol{x}}_t^\top\big]^\top \in \mathbb{R}^{DC}$. The problem is formulated as the minimization of the cost

$$\begin{aligned} c &= \big(\hat{\boldsymbol{\zeta}}_T - \boldsymbol{\zeta}_T\big)^\top \boldsymbol{Q}_T \big(\hat{\boldsymbol{\zeta}}_T - \boldsymbol{\zeta}_T\big) \\ &\quad + \sum_{t=1}^{T-1} \Big(\big(\hat{\boldsymbol{\zeta}}_t - \boldsymbol{\zeta}_t\big)^\top \boldsymbol{Q}_t \big(\hat{\boldsymbol{\zeta}}_t - \boldsymbol{\zeta}_t\big) \;+\; \boldsymbol{u}_t^\top \boldsymbol{R}_t\, \boldsymbol{u}_t \Big) \\ &= \big(\boldsymbol{\mu_s} - \boldsymbol{\zeta}\big)^\top \boldsymbol{Q_s} \big(\boldsymbol{\mu_s} - \boldsymbol{\zeta}\big) \;+\; \boldsymbol{u}^\top \boldsymbol{R_s} \boldsymbol{u}, \end{aligned} \tag{47}$$

with $\boldsymbol{\zeta} = \big[\boldsymbol{\zeta}_1^\top, \boldsymbol{\zeta}_2^\top, \ldots, \boldsymbol{\zeta}_T^\top\big]^\top \in \mathbb{R}^{DCT}$ the evolution of the state variable and $\boldsymbol{u} = \big[\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \ldots, \boldsymbol{u}_{T-1}^\top\big]^\top \in \mathbb{R}^{D(T-1)}$ the evolution of the control variable. $\boldsymbol{\mu_s} = \big[\hat{\boldsymbol{\zeta}}_{s_1}^\top, \hat{\boldsymbol{\zeta}}_{s_2}^\top, \ldots, \hat{\boldsymbol{\zeta}}_{s_T}^\top\big]^\top \in \mathbb{R}^{DCT}$ represents the evolution of the tracking target. $\boldsymbol{Q_s} = \text{blockdiag}(\boldsymbol{Q}_{s_1}, \boldsymbol{Q}_{s_2}, \ldots, \boldsymbol{Q}_{s_T}) \in \mathbb{R}^{DT\times DT}$ represents the evolution of the required tracking precision, and $\boldsymbol{R_s} = \text{blockdiag}(\boldsymbol{R}_{s_1}, \boldsymbol{R}_{s_2}, \ldots, \boldsymbol{R}_{s_{T-1}}) \in \mathbb{R}^{D(T-1)\times D(T-1)}$ represents the evolution of the cost on the control inputs. The problem corresponds to an unconstrained linear *model predictive control* (MPC) problem. It is worth noting that the objective function used in the context of trajectory-GMM (see Section 3.6.2) is similar to the cost function in (47) without control cost (i.e., with $\boldsymbol{R_s} = \boldsymbol{0}$).

The tracking problem can be solved by different techniques, either exploiting tools from physics, dynamic programming or linear algebra [7]. It can, for example, be solved with a batch approach and simple linear algebra, by expressing all future states $\boldsymbol{\zeta}_t$ as an explicit function of the state $\boldsymbol{\zeta}_1$. By writing

$$\begin{aligned} \boldsymbol{\zeta}_2 &= \boldsymbol{A}\boldsymbol{\zeta}_1 + \boldsymbol{B}\boldsymbol{u}_1, \\ \boldsymbol{\zeta}_3 &= \boldsymbol{A}\boldsymbol{\zeta}_2 + \boldsymbol{B}\boldsymbol{u}_2 = \boldsymbol{A}(\boldsymbol{A}\boldsymbol{\zeta}_1 + \boldsymbol{B}\boldsymbol{u}_1) + \boldsymbol{B}\boldsymbol{u}_2, \\ &\;\vdots \\ \boldsymbol{\zeta}_T &= \boldsymbol{A}^{T-1}\boldsymbol{\zeta}_1 + \boldsymbol{A}^{T-2}\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{A}^{T-3}\boldsymbol{B}\boldsymbol{u}_2 + \cdots + \boldsymbol{B}\boldsymbol{u}_{T-1}, \end{aligned}$$

in a matrix form, we get

$$\underbrace{\begin{bmatrix} \boldsymbol{\zeta}_1 \\ \boldsymbol{\zeta}_2 \\ \boldsymbol{\zeta}_3 \\ \vdots \\ \boldsymbol{\zeta}_T \end{bmatrix}}_{\boldsymbol{\zeta}} = \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{T-1} \end{bmatrix}}_{\boldsymbol{S^\zeta}} \boldsymbol{\zeta}_1 + \underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{T-2}\boldsymbol{B} & \boldsymbol{A}^{T-3}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{S^u}} \underbrace{\begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_{T-1} \end{bmatrix}}_{\boldsymbol{u}}, \tag{48}$$

with $\boldsymbol{\zeta} \in \mathbb{R}^{DCT}$, $\boldsymbol{S^\zeta} \in \mathbb{R}^{DCT\times DC}$, $\boldsymbol{\zeta}_1 \in \mathbb{R}^{DC}$, $\boldsymbol{S^u} \in \mathbb{R}^{DCT\times D(T-1)}$ and $\boldsymbol{u} \in \mathbb{R}^{D(T-1)}$. Substituting (48) into (47), we get the cost function

$$c = \big(\boldsymbol{\mu_s} - \boldsymbol{S^\zeta}\boldsymbol{\zeta}_1 - \boldsymbol{S^u}\boldsymbol{u}\big)^\top \boldsymbol{Q_s} \big(\boldsymbol{\mu_s} - \boldsymbol{S^\zeta}\boldsymbol{\zeta}_1 - \boldsymbol{S^u}\boldsymbol{u}\big) + \boldsymbol{u}^\top \boldsymbol{R_s}\boldsymbol{u}. \tag{49}$$

Differentiating with respect to $\boldsymbol{u}$ and equating to zero yield the sequence of control inputs

$$\hat{\boldsymbol{u}} = \big(\boldsymbol{S^u}^\top \boldsymbol{Q_s} \boldsymbol{S^u} + \boldsymbol{R_s}\big)^{-1} \boldsymbol{S^u}^\top \boldsymbol{Q_s}\, \big(\boldsymbol{\mu_s} - \boldsymbol{S^\zeta}\boldsymbol{\zeta}_1\big), \tag{50}$$

corresponding to a weighted least squares estimate with Tikhonov regularization (ridge regression); see also Section 3.6.

Similarly to the trajectory-GMM described in Section 3.6.2, the error on the ridge regression estimate can be used to compute a covariance $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{u}}$ in control space. By using the linear relation in (48), the distribution $\mathcal{N}(\hat{\boldsymbol{u}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{u}})$ in control space can then be converted to a distribution $\mathcal{N}(\hat{\boldsymbol{\zeta}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{\zeta}})$ in feature space with parameters

$$\hat{\boldsymbol{\zeta}} = \boldsymbol{S^\zeta}\boldsymbol{\zeta}_1 + \boldsymbol{S^u}\hat{\boldsymbol{u}}, \tag{51}$$

$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{\zeta}} = \sigma \boldsymbol{S^u} \big(\boldsymbol{S^u}^\top \boldsymbol{Q_s} \boldsymbol{S^u} + \boldsymbol{R_s}\big)^{-1} \boldsymbol{S^u}^\top. \tag{52}$$

The controller in (50) can alternatively be retrieved iteratively by relying on dynamic programming or the Pontryagin maximization principle. For the discrete version of the dynamical system defined by (46), two costate variables $\boldsymbol{P}_t$ and $\boldsymbol{d}_t$ are introduced, and the cost is optimized by recursion with

$$\boldsymbol{P}_t = \boldsymbol{Q}_t - \boldsymbol{A}^\top \Big(\boldsymbol{P}_{t+1}\boldsymbol{B}\, \big(\boldsymbol{B}^\top \boldsymbol{P}_{t+1} \boldsymbol{B} + \boldsymbol{R}_t\big)^{-1} \boldsymbol{B}^\top \boldsymbol{P}_{t+1} - \boldsymbol{P}_{t+1} \Big) \boldsymbol{A}, \tag{53}$$

$$\begin{aligned} \boldsymbol{d}_t = \Big(\boldsymbol{A}^\top - \boldsymbol{A}^\top \boldsymbol{P}_{t+1} \boldsymbol{B} \big(\boldsymbol{B}^\top \boldsymbol{P}_{t+1} \boldsymbol{B} + \boldsymbol{R}_t\big)^{-1} \boldsymbol{B}^\top \Big) \\ \Big(\boldsymbol{P}_{t+1} (\boldsymbol{A}\hat{\boldsymbol{\zeta}}_t - \hat{\boldsymbol{\zeta}}_{t+1}) + \boldsymbol{d}_{t+1} \Big), \end{aligned} \tag{54}$$

which are solved backward in time from the terminal conditions set by $\boldsymbol{P}_T = \boldsymbol{Q}_T$ and $\boldsymbol{d}_T = \boldsymbol{0}$. The first equation is a Riccati equation for the discrete formulation of LQR, while the second is a linear differential equation used to compute the feedforward term (which depends on the solution of the Riccati equation).

The costate variables $\boldsymbol{P}_t$ and $\boldsymbol{d}_t$ are then used to compute the control commands $\boldsymbol{u}_t$ using the forward integration

$$\boldsymbol{u}_t = \boldsymbol{K}_t(\hat{\boldsymbol{\zeta}}_t - \boldsymbol{\zeta}_t) + \boldsymbol{f}_t, \tag{55}$$

with feedback gain and feedforward terms defined as

$$\boldsymbol{K}_t = (\boldsymbol{B}^{\top}\boldsymbol{P}_t\boldsymbol{B} + \boldsymbol{R}_t)^{-1}\boldsymbol{B}^{\top}\boldsymbol{P}_t\boldsymbol{A}, \tag{56}$$

$$\boldsymbol{f}_t = -(\boldsymbol{B}^{\top}\boldsymbol{P}_t\boldsymbol{B} + \boldsymbol{R}_t)^{-1}\boldsymbol{B}^{\top}\big(\boldsymbol{P}_t(\boldsymbol{A}\hat{\boldsymbol{\zeta}}_t - \hat{\boldsymbol{\zeta}}_t) + \boldsymbol{d}_t\big). \tag{57}$$

Alternatively, the tracking problem can be recast as a regulation problem (namely, with a constant target at zero) by considering a dynamical system with augmented state defined by

$$\underbrace{\begin{bmatrix}\boldsymbol{\zeta}_{t+1}\\ 1\end{bmatrix}}_{\tilde{\boldsymbol{\zeta}}_{t+1}} = \underbrace{\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{0} & 1\end{bmatrix}}_{\tilde{\boldsymbol{A}}}\underbrace{\begin{bmatrix}\boldsymbol{\zeta}_t\\ 1\end{bmatrix}}_{\tilde{\boldsymbol{\zeta}}_t} + \underbrace{\begin{bmatrix}\boldsymbol{B}\\ \boldsymbol{0}\end{bmatrix}}_{\tilde{\boldsymbol{B}}}\boldsymbol{u}_t, \tag{58}$$

together with the augmented tracking weight

$$\tilde{\boldsymbol{Q}}_t = \begin{bmatrix}\boldsymbol{Q}_t^{-1}+\hat{\boldsymbol{\zeta}}_t\hat{\boldsymbol{\zeta}}_t^{\top} & \hat{\boldsymbol{\zeta}}_t\\ \hat{\boldsymbol{\zeta}}_t^{\top} & 1\end{bmatrix}^{-1}, \tag{59}$$

which is used to define the cost

$$c = \tilde{\boldsymbol{\zeta}}_T^{\top}\tilde{\boldsymbol{Q}}_T\tilde{\boldsymbol{\zeta}}_T \;+\; \sum_{t=1}^{T-1}\Big(\tilde{\boldsymbol{\zeta}}_T^{\top}\tilde{\boldsymbol{Q}}_t\tilde{\boldsymbol{\zeta}}_T \;+\; \boldsymbol{u}_t^{\top}\boldsymbol{R}_t\boldsymbol{u}_t\Big), \tag{60}$$

optimized by recursion with

$$\boldsymbol{P}_t = \tilde{\boldsymbol{Q}}_t - \tilde{\boldsymbol{A}}^{\top}\Big(\boldsymbol{P}_{t+1}\tilde{\boldsymbol{B}}\;(\tilde{\boldsymbol{B}}^{\top}\boldsymbol{P}_{t+1}\tilde{\boldsymbol{B}}+\boldsymbol{R}_t)^{-1}\tilde{\boldsymbol{B}}^{\top}\boldsymbol{P}_{t+1} - \boldsymbol{P}_{t+1}\Big)\tilde{\boldsymbol{A}}, \tag{61}$$

solved backward in time from the terminal conditions set by $\boldsymbol{P}_T = \tilde{\boldsymbol{Q}}_T$.

$\boldsymbol{P}_t$ is then used to compute the control commands $\boldsymbol{u}_t$ using the forward integration

$$\boldsymbol{u}_t = -\tilde{\boldsymbol{K}}_t\,\tilde{\boldsymbol{\zeta}}_t, \tag{62}$$

with a feedback gain defined as

$$\tilde{\boldsymbol{K}}_t = (\tilde{\boldsymbol{B}}^{\top}\boldsymbol{P}_t\tilde{\boldsymbol{B}} + \boldsymbol{R}_t)^{-1}\tilde{\boldsymbol{B}}^{\top}\boldsymbol{P}_t\tilde{\boldsymbol{A}}. \tag{63}$$

A similar approach can be used by defining the system in a continuous form instead of discrete form; see, e.g., [7]. In addition, an infinite time horizon can be considered instead of the finite horizon as presented in the above (also both in discrete and continuous form).

The conventional use of the above technique in control is to let the experimenter define the weighting terms $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ to find a controller to track a predetermined target or trajectory $\hat{\boldsymbol{\zeta}}_t$. When determined by the experimenter, $\boldsymbol{Q}_t$ and $\boldsymbol{R}_t$ are most often defined as diagonal and constant in time. $\boldsymbol{Q}_t$ can also typically be defined by the experimenter as being null for a range of time steps (e.g., to indicate with $\boldsymbol{Q}_t$ the need to pass through a set of via-points).

Learning the weights in the objective function (47) can be viewed as a basic form of inverse optimal control [1]. Several approaches have been proposed to exploit the above formulation within a learning control context. In [73], the ProMP representation (see §3.6.1) is exploited within a dedicated LQT control formulation exploiting the property of the covariance derivatives that can be computed explicitly, instead of using a backward integration procedure as in the conventional algorithm. In [65], the above approach is used in the context of risk-sensitive control for haptic assistance. This is done by exploiting the predicted variability to build a controller for the robot (in task space or in joint space). The retrieved variability and correlation information is first learned from the provided data and is then exploited to generate safe and natural movements within an optimal control strategy, in accordance to the predicted range of motion that can be exploited to reproduce the task in the current situation.

The approach can be generalized to cost functions acting simultaneously in multiple coordinate systems [9, 104]. In this case, a task-parameterized Gaussian mixture model (TP-GMM) is used to estimate from demonstrations the trajectory to track (defining $\hat{\boldsymbol{\zeta}}_t$), as well as the required tracking accuracy and coordination (defining $\boldsymbol{Q}_t$). In this application, both $\hat{\boldsymbol{\zeta}}_t$ and $\boldsymbol{Q}_t$ are expressed in several coordinate systems and are changing with time, with $\boldsymbol{Q}_t$ describing full precision matrices. The approach is used to learn a controller from demonstrations with time-varying proportional and derivative gains, also known as gain scheduling. The resulting controller can adapt to new situations by autonomously adapting both the reference trajectory and the impedance behavior. This autonomous regulation of the stiffness and damping based on the variations observed in the demonstrations provides a *minimal intervention control* strategy in which deviations from the retrieved average trajectory are corrected only when they interfere with task performance [96]. In humanoids, such control strategy can be useful for reducing energy consumption, as well as for safety when collaborating with users or when contacts with the environment occur.

Figures 12 and 13 present examples of LQT controllers with a trajectory distribution retrieved by HSMM. The second column of Fig. 4 also shows an example of the retrieved flow field.

4.2 Reward-weighted optimization

We have seen in the previous sections that motion skills and controllers can be represented in several compact parametric forms. We will represent a controller in this parameter space as $\boldsymbol{\theta}_n$ and will discuss in this section how the controller can be refined iteratively by self-refinement. Since this chapter does not aim at covering the whole field of reinforcement learning, we will narrow the survey to a recent trend in reinforcement learning to use parameterized policies in combination with probability-weighted averaging.

Several reward-weighted optimization mechanisms are

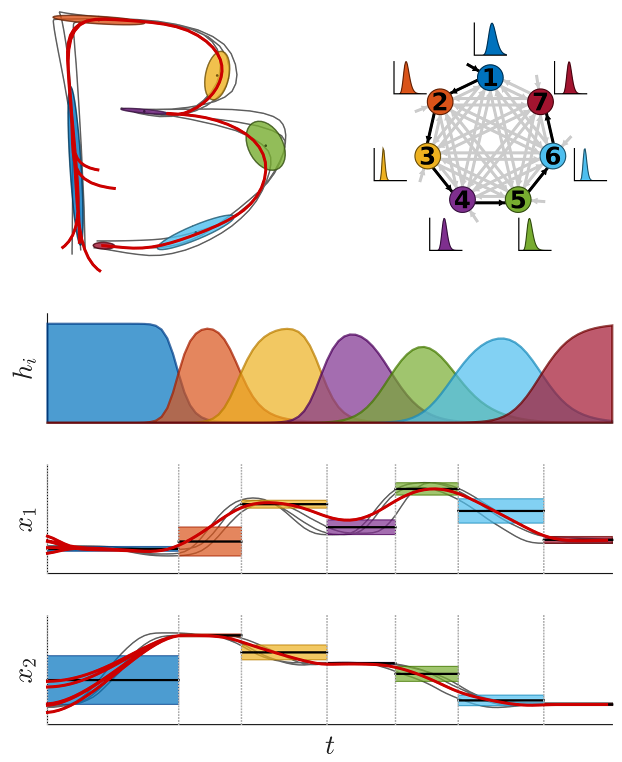


Figure 12: Linear quadratic tracking (LQT) with weights and target sequence generated from an HSMM. *Top-left:* Demonstrations of a movement (in gray lines), HSMM distribution outputs (ellipses with a different color for each state), and five reproduction attempts (in red lines) starting from initial points in the vicinity of the demonstrations. *Top-right:* HSMM transition graph and state duration as lognormal distributions, with the seven states depicted with distinct colors. *Bottom:* The timeline graphs at the bottom show the evolution of the activation weights and the planar movement (demonstrations and reproductions in gray and red lines, respectively). The colored blocks represent the stepwise transitions of the different HSMM states, resulting in smooth trajectories after LQT.

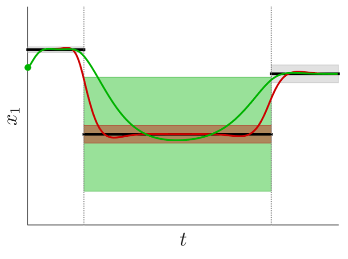


Figure 13: An HSMM can be used to generate a sequence of states, each characterized by a center and a covariance. The sequence of centers is used as a stepwise reference (black line segments), and the full covariance matrices are used to build a sequence of full precision matrices (shaded areas around the line segments). When combined with LQT, the stepwise reference is smoothly tracked (red and green lines) from any initial starting point (red and green points). We can see that the observed variations and coordination patterns influence the controller. Here, an observed invariant motion segment (red shaded area) will be more aggressively tracked than a segment allowing variations (green shaded area).

based on iteratively repeating the procedure

$$\begin{aligned}
&\boldsymbol{\theta}_n \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\theta}}, \boldsymbol{\Sigma}^{\boldsymbol{\theta}}) \quad \forall n \in 1,\dots,N_s,\\
&\boldsymbol{\theta}_n \leftarrow \text{sort}(\boldsymbol{\theta}_n) \quad \text{w.r.t. } r_n = r(\boldsymbol{\theta}_n),\\
&\boldsymbol{\Sigma}^{\boldsymbol{\theta}} \leftarrow \sum_{n=1}^{N_s} w_n (\boldsymbol{\theta}_n - \boldsymbol{\mu}^{\boldsymbol{\theta}})(\boldsymbol{\theta}_n - \boldsymbol{\mu}^{\boldsymbol{\theta}})^{\top},\\
&\boldsymbol{\mu}^{\boldsymbol{\theta}} \leftarrow \sum_{n=1}^{N_s} w_n \boldsymbol{\theta}_n, \qquad (64)
\end{aligned}$$

where $r(\boldsymbol{\theta}_n)$ is a given reward function evaluated by applying the controller defined by its parameters $\boldsymbol{\theta}_n$ stored in a vector form, and N_s is the number of samples. $r(\boldsymbol{\theta}_n)$ is often defined in a Gaussian, RBF, or exponential form to obtain a similar update mechanism, as in an expectation-maximization procedure. $\boldsymbol{\Sigma}^{\boldsymbol{\theta}}$ indicates where to search at each iteration and $\boldsymbol{\mu}^{\boldsymbol{\theta}}$ indicates the estimated average solution in the parameter space.

In CEM (cross-entropy method) [50], an elite update mechanism is used with $w_n = \frac{1}{N_e}$ for the first N_e samples and $w_n = 0$ otherwise. In PoWER (Policy Learning by Weighting Exploration with the Returns) [45], a reward-weighted update is used with $w_n = \frac{r_n}{\sum_{m=1}^{N_s} r_m}$, and the update is equivalently rewritten as $\boldsymbol{\mu}^{\boldsymbol{\theta}} \leftarrow \boldsymbol{\mu}^{\boldsymbol{\theta}} + \sum_{n=1}^{N_s} w_n(\boldsymbol{\theta}_n - \boldsymbol{\mu}^{\boldsymbol{\theta}})$ to make connections with gradient-based approaches in reinforcement learning and to show that the update is within a convex hull of the sampled trials, providing a conservative update rule. The collection of CMA-ES (covariance matrix adaptation evolution strategy) approaches [39] exploits a similar form of updates, but they use a more elaborated form of covariance update considering a history (evolution path). In [14], the procedure in (64) is extended to a mixture model by using a potentially growing number of Gaussians during exploration to cope with multiple control options arising during the search.

PI^2 (policy improvement with path integrals) [8] has been derived from a different framework (stochastic optimal control) but also results in a probability-weighted averaging procedure with parametrized policy, by defining a reward function that can be evaluated at each time step based on the retrieved trajectory. The cost of a trajectory is determined by evaluating the reward for each time step t, with a different parameter update $\boldsymbol{\mu}^{\boldsymbol{\theta}}_t$. A single parameter update is then computed by averaging over all time steps, weighted such that earlier updates contribute more (since earlier updates affect a larger time horizon, they have more influence). In [87], PI^2 is linked to the techniques described above to update the sampling covariance guiding exploration. The resulting PI^2-CMA formulation can be summarized by the iterative procedure

$$\begin{aligned}
&\boldsymbol{\theta}_n \sim \mathcal{N}(\boldsymbol{\mu}^{\boldsymbol{\theta}}, \boldsymbol{\Sigma}^{\boldsymbol{\theta}}) \quad \forall n \in 1,\dots,N_s,\\
&\left.\begin{aligned}
\boldsymbol{\Sigma}^{\boldsymbol{\theta}}_t &\leftarrow \textstyle\sum_{n=1}^{N_s} w_{n,t}(\boldsymbol{\theta}_n - \boldsymbol{\mu}^{\boldsymbol{\theta}})(\boldsymbol{\theta}_n - \boldsymbol{\mu}^{\boldsymbol{\theta}})^{\top}\\
\boldsymbol{\mu}^{\boldsymbol{\theta}}_t &\leftarrow \textstyle\sum_{n=1}^{N_s} w_{n,t}\boldsymbol{\theta}_n
\end{aligned}\right\} \forall t \in 1,\dots,T,\\
&\boldsymbol{\Sigma}^{\boldsymbol{\theta}} \leftarrow \textstyle\sum_{t=1}^{T} w_t \boldsymbol{\Sigma}^{\boldsymbol{\theta}}_t, \qquad \boldsymbol{\mu}^{\boldsymbol{\theta}} \leftarrow \textstyle\sum_{t=1}^{T} w_t \boldsymbol{\mu}^{\boldsymbol{\theta}}_t, \qquad (65)
\end{aligned}$$

with $w_{n,t} = \frac{r_{n,t}}{\sum_{m=1}^{N_s} r_{m,t}}$ and $w_t = \frac{T-t}{\sum_{s=1}^{T} T-s}$. The reward function $r_{n,t}$ is usually defined as a cost-to-go function in exponential form.

The early developments in PI^2 paved the way to several improvements or variants of the original approach, including policy improvement through black box optimization (PI$^{\mathrm{BB}}$) [88], path integral relative-entropy policy search (PI-REPS) [27], or reward optimization with compact kernels and fast natural gradient regression (ROCK*) [35].

Applications of reward-weighted optimization approaches include skills such as controlling the center of mass in biped walking [49], standing up from a chair [28], flipping pancakes [14], propelling arrows [47], manipulating a rod with two hands [91] or pushing/opening a door [90].

4.3 Iterative learning control (ILC)

Iterative learning control (ILC) is another paradigm to refine a controller by self-practice. The goal of ILC and *repetitive control* (RC) is to improve the performance of control systems by adjusting the commands through learning from previous control trials [6]. They are typically used for repetitive or cyclic tasks. ILC and RC resemble the human learning process (practice of a task) which updates control input based on error signals from previous trials.

The formulation of ILC assumes deterministic system dynamics, repeatability of the target tracking task over a finite time horizon, and the same initial conditions of each trial. Assume we have a tracking task with control input u and output error $e = y_d - y$, a widely used ILC learning algorithm is formulated as

$$u_{i+1}(t) = u_i(t) + k_l e_i(t), \quad t \in [0, T_{\text{iter}}], \tag{66}$$

where the subscript i denotes the iteration number, T_{iter} is the period of one iteration, and k_l is the learning gain which defines the learning speed. The learning time variable t is reset to 0 at the beginning of each iteration.

There are several variants of ILC. For example, the discrete-time PD-type ILC update can be written as

$$u_{i+1}(t) = u_i(t) + k_{lp} e_i(t) + k_{ld}\big(e_i(t) - e_i(t-1)\big), \tag{67}$$

where k_{lp} is the proportional gain and k_{ld} is the derivative gain. The ILC update law can be written as

$$u_{i+1}(t) = u_0(t) + k_f\big(u_i(t) - u_0(t)\big) + k_l e_i(t), \tag{68}$$

where $0 < k_f \leq 1$ is a forgetting factor. The convergence condition of (68) is given as

$$|k_f - k_l d| < 1. \tag{69}$$

A forgetting factor smaller than one robustifies the ILC but decreases the performance since the learned useful information is also discounted.

A feedback controller can handle unknown disturbances and uncertainties in the system model, but has a lag in transient tracking. In contrast, ILC is anticipatory and generates an open-loop control signal through feedback in the iteration domain. ILC learns compensation terms for repeating noise and disturbance. In order to respond to unanticipated and non-repeating disturbances, ILC is in practice often combined with a feedback controller, namely,

$$u_{i+1}(t) = u_i(t) + k_l e_i(t) + k e_{i+1}(t), \quad t \in [0, T_{\text{iter}}]. \tag{70}$$

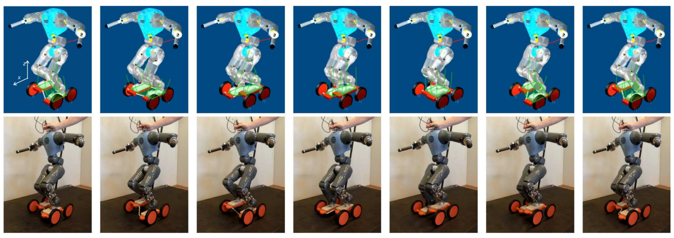


Figure 14: One period of COMAN steady-state pedaling on a pedal racer in *Webots* dynamic simulator (top) and in experiment (bottom).

In early work, ILC was most often used for tracking of repetitive tasks with industrial manipulators. Recently, ILC started to be incorporated in programming by demonstration for humanoid robots. In [25], a discrete dynamical movement primitives (DMP) was modulated for interaction with environment by ILC. While a typical DMP consists of a second-order linear dynamical system with a linear combination of nonlinear radial basis functions (as shown in Section 3.3), an ILC-modulated DMP introduces an additional nonlinear modulation term which is learned by ILC from force/torque feedback.

Repetitive control (RC) is closely related to ILC [63]. Instead of making repeated runs of a desired finite time trajectory, RC aims to perfectly execute a periodic command or to execute a periodic command in the presence of a periodic disturbance. Repetitive control was, for example, employed on a task of riding a pedal racer which demands balancing of a humanoid robot [24]; see also Fig. 14. In this work, periodic dynamical movement primitives are combined with RC according to force feedback.

When it comes to gait control in humanoid robots, it seems reasonable to ask whether the model uncertainties of the complex multi-body dynamics could be compensated by incorporating error information from previous trials (by trial-error learning). But as pointed out in [63], the original form of ILC or RC cannot be directly applicable to the gait problem. The dynamic models for walking motions are highly nonlinear, and stability of the bipedal walking is nontrivial. Moreover, unlike the pedaling example, bipedal locomotion includes jump discontinuities, and the duration of the phases is not the same from cycle to cycle. In [63], four RC laws were proposed for bipedal gait and showed reduced tracking errors in joint space in simulation.

Although the feasibility of balancing on a pedal racer was demonstrated with the COMAN humanoid robot in [24], there are additional challenges in bipedal walking. In the former case, the robot's feet keep the contact with the pedal racer during riding, and the feet trajectory is limited to a motion primitive constrained by the kinematics of the pedal. In contrast, foot contact with the ground involves discontinuities requiring walking motions to be represented by a sequence of different repetitive motions.

Hu *et al.* [34] proposed an online iterative learning control of zero-moment point (ZMP) trajectory for biped walking, with the focus of improving walking robustness by refining the walking pattern, i.e., reducing the effect of unmodeled dynamics in the pattern generation stage. The key idea is to learn a feedforward compensative ZMP

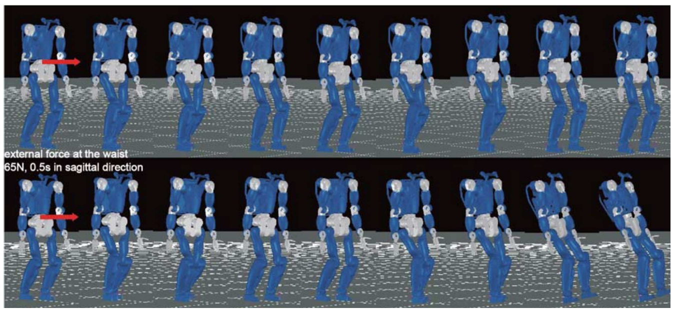


Figure 15: Snapshots of short period disturbance simulation recorded from 22s and with 0.4s time interval. Upper row shows the results with learning and lower row shows the result without learning. The learned compensative term shows the improved stability during walking.

(CZMP) term through the actual ZMP error during the repetition of the walking trials. The learned CZMP adjusts the ZMP reference trajectory and reduces the ZMP disturbances due to the unmodeled dynamics, making the measured robot ZMP converge to the desired ZMP trajectory. This feedforward term is added to the conventional ZMP-based online walking controllers of the preview control-based pattern generator and the online feedback balancer. Since the walking is only repetitive in the local coordinate (with respect to the foot stance coordinate at each walking cycle), the learning process is applied in the local coordinate system of each iteration. The learning process is conducted continuously without reset between successive iterations. In order to achieve the continuity of the learning process, an initialization iteration is designed to transit smoothly from non-learning to learning phases. The improved walking robustness was shown in simulations and experiments. Figure 15 depicts a simulation result showing the stability of the DLR TORO humanoid robot against external disturbances during walking with and without learning. Improved convergence speed and reduced ZMP error during transition phases between different walking motions are achieved.

4.4 Task priority learning

An important and challenging category of applications in learning control for humanoid robots concerns the learning of priority constraints. It relates to the challenge of organizing movement primitives not only in series but also in parallel (both for recognition and synthesis). The problem of learning and controlling a humanoid by considering task priorities can be tackled both at kinematic and dynamic levels. We will define it here at a kinematic level with a robot controlled with joint angle velocity commands, but the presented techniques can be extended to torque-based controllers.

We start from the objective function (1) used in Section 3.1 for computing a single least norm estimate. We now consider the general solution of this linear system, which is given by

$$\hat{\boldsymbol{A}} = \boldsymbol{X}^{\mathcal{I}\dagger}\boldsymbol{X}^{\mathcal{O}} + \overbrace{(\boldsymbol{I} - \boldsymbol{X}^{\mathcal{I}\dagger}\boldsymbol{X}^{\mathcal{I}})}^{\boldsymbol{N}}\boldsymbol{V}, \tag{71}$$

where the right pseudoinverse of $\boldsymbol{X}$ is defined by $\boldsymbol{X}^\dagger = \boldsymbol{X}^{\mathcal{I}\top}(\boldsymbol{X}^{\mathcal{I}}\boldsymbol{X}^{\mathcal{I}\top})^{-1}$, $\boldsymbol{N}$ is a *nullspace projection operator*, and $\boldsymbol{V}$ can be any vector/matrix (e.g., resulting from the minimization of a secondary objective function). In the above, the solution is unique if and only if $\boldsymbol{A}$ has full column rank, in which case $\boldsymbol{N}$ is a zero matrix. The nullspace projection guarantees that $\|\boldsymbol{X}^{\mathcal{O}}-\boldsymbol{X}^{\mathcal{I}}\hat{\boldsymbol{A}}\|_2$ is still minimized. An alternative way of computing the nullspace projection matrix is to exploit the singular value decomposition $\boldsymbol{X}^{\mathcal{I}\dagger}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ to compute $\boldsymbol{N} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{U}}^\top$, where $\tilde{\boldsymbol{U}}$ is a matrix formed by the columns of $\boldsymbol{U}$ that span for the corresponding zero rows in $\boldsymbol{\Sigma}$.

The general solution above is important when considering the control of redundant kinematic chains such as in humanoid robots, when $D^{\mathcal{O}} < D^{\mathcal{I}}$. Indeed, humanoids are often required to produce movements described in task space while being controlled in joint space. The problem is commonly formulated with an objective function similar to (1), by first computing the forward kinematics relationship as

$$\boldsymbol{x}_t = f(\boldsymbol{q}_t) \iff \dot{\boldsymbol{x}}_t = \frac{\partial \boldsymbol{x}_t}{\partial t} = \frac{\partial f(\boldsymbol{q}_t)}{\partial \boldsymbol{q}_t}\frac{\partial \boldsymbol{q}_t}{\partial t} = \boldsymbol{J}(\boldsymbol{q}_t)\,\dot{\boldsymbol{q}}_t, \tag{72}$$

where $\boldsymbol{J}(\boldsymbol{q}_t) = \frac{\partial f(\boldsymbol{q}_t)}{\partial \boldsymbol{q}_t}$ is a Jacobian matrix, and by computing inverse kinematics as the least squares solution

$$\hat{\dot{\boldsymbol{q}}}_t = \boldsymbol{J}^\dagger(\boldsymbol{q}_t)\,\dot{\boldsymbol{x}}_t + \boldsymbol{N}(\boldsymbol{q}_t)\,g(\boldsymbol{q}_t). \tag{73}$$

Humanoids are frequently required to handle multiple tasks in parallel that can be conflicting. This, for example, corresponds to the control of the two hands and feet in contact with the environment or the control of the center of mass for balancing, requiring the definition of multiple Jacobians for the different locations of interest in the kinematic chain.

Learning control in humanoids requires the handling of such task prioritization, where the goal is to learn how to handle the priorities of multiple tasks running in parallel (from demonstration or self-refinement). Early approaches can be categorized in two research directions, by either exploiting a strict hierarchy structure, as a generalization of (73) applied to multilevel hierarchies, or by employing a weighted least squares solution as in (5) for the inverse kinematics.

The two techniques have pros and cons. Setting an explicit nullspace structure guarantees strict priorities at the expense of constraining sometimes too much the tasks, which quickly limits the number of tasks that can simultaneously be handled, compared to the number of degrees of freedom available for controlling the robot. It can also create discontinuities in the control problem when switching from one hierarchy structure to another. A soft weighting scheme can in contrast handle different levels of task importance and gradual changes from one task to another, but it does not provide strict guarantees on the fulfillment of each separated task; see Fig. 16 for an illustrative example.

Because of the limitations in selecting one or the other approach, researchers have proposed alternative solutions aiming to gather the benefit of the two techniques, which can be divided in four broad categories, with approaches concentrating on:

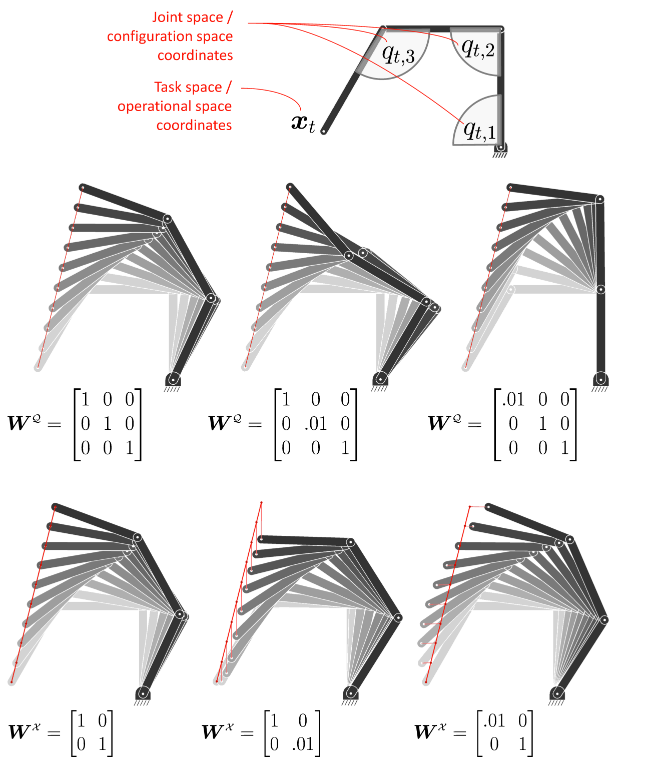

Joint space / configuration space coordinates
Task space / operational space coordinates
$q_{t,3}$
$q_{t,2}$
$q_{t,1}$
$\boldsymbol{x}_t$
$\boldsymbol{W}^{\mathcal{Q}} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$
$\boldsymbol{W}^{\mathcal{Q}} = \begin{bmatrix}1&0&0\\0&.01&0\\0&0&1\end{bmatrix}$
$\boldsymbol{W}^{\mathcal{Q}} = \begin{bmatrix}.01&0&0\\0&1&0\\0&0&1\end{bmatrix}$
$\boldsymbol{W}^{x} = \begin{bmatrix}1&0\\0&1\end{bmatrix}$
$\boldsymbol{W}^{x} = \begin{bmatrix}1&0\\0&.01\end{bmatrix}$
$\boldsymbol{W}^{x} = \begin{bmatrix}.01&0\\0&1\end{bmatrix}$

Figure 16: Task prioritization based on a weighted least squares estimate of inverse kinematics. *Top:* Variables involved. *Middle:* with weights in joint space (with right pseudoinverse $\hat{\dot{\boldsymbol{q}}}_t = \boldsymbol{W}^{\mathcal{Q}}\boldsymbol{J}^{\top}(\boldsymbol{J}\boldsymbol{W}^{\mathcal{Q}}\boldsymbol{J}^{\top})^{-1}\,\dot{\boldsymbol{x}}_t$; see also (6)). *Bottom:* with weights in task space (with left pseudoinverse $\hat{\dot{\boldsymbol{q}}}_t = (\boldsymbol{J}^{\top}\boldsymbol{W}^{x}\boldsymbol{J})^{-1}\boldsymbol{J}^{\top}\boldsymbol{W}^{x}\,\dot{\boldsymbol{x}}_t$; see also (5)).

1. Developing more versatile representations of task prioritization;
2. Learning task priority from a pool of candidates, either from demonstration or by self-refinement;
3. Learning the nullspace projection operators;
4. Developing human-robot teaching techniques that can exploit already acquired nullspace structures.

Some of the work in the first research direction listed above do not directly tackle a learning problem, but they provide representations that facilitate task priority learning, with the aim of recasting the problem as a more standard task learning problem. [20] is an example of such approach, where a continuous nullspace projection technique is developed to consider unilateral constraints, singular Jacobian matrices, and dynamic variations of the priority order within the hierarchy structure. Inspired by this approach, [60] propose a generalized projector to extend the singular value decomposition used in [20] to compute the nullspace projection operator, so that the generalized projector can allow a task to be completely, partially, or never projected into the nullspace of other tasks, with a continuous priority parameterization. In this framework, a strict priority becomes a limit case of a relative priority of tasks, which is useful for humanoids acting in dynamically changing contexts, since task priorities may have to be switched in order to cope with changing situations or when non-strict priorities between tasks may become strict ones. Task hierarchies are handled by the modulation of a priority matrix, without necessarily modifying the control formulation each time the hierarchy changes. Typically, the evolution of the task priorities is designed manually with weight functions ranging from 0 to 1, controlling if each task is fully, partially, or never projected in the nullspaces of the other tasks with higher priority. [67] is another example, where a weighted sum of torques, each minimizing a different cost function, provides a convenient representation but requires the mixing coefficients to be manually selected. In [2], a description of multiple task definitions is constructed with the concept of flexible priority structures. It handles efficiently unprioritized or prioritized accumulations of tasks and priority switches by incorporating interpolation in joint space. Consequently, smooth, arbitrary, and consecutive task transitions are achieved.

In parallel to the techniques described above, a collection of work concentrates on learning task priority either by stochastic search or from demonstration. Most of the techniques assume that the set of elementary tasks is known but can be used in conjunction with learning techniques to acquire the elementary tasks. In [18], a mixture of torque controllers is employed, with the mixing coefficients learned by stochastic search with CMA-ES (a derivative-free optimization strategy; see Section 4.2). In [66], task priorities are learned by encoding the temporal profile of the mixing coefficients with a RBF decomposition of the signal and by using CMA-ES as stochastic search algorithm. In [61], the presence of interferences between multiple tasks encoded in DMPs is detected, where the tasks are then iteratively modified to resolve the potential interferences. This is achieved by stochastic optimization in the DMP parameters space. The approach exploits the fact that often, tasks are described within an acceptable range of error, providing an opportunity to render them compatible without an explicit need for prioritization. In [62], a technique based on GPR is developed to exploit task variability as a way to modulate task priorities during execution, by temporarily deviating certain tasks as needed in the presence of incompatibilities. In [53], an optimization framework with a nested sequence of objectives (so as not to conflict with higher-priority objectives) is proposed with the aim of unifying prioritized task-space and optimization-based control. For the case of positive semidefinite quadratic objectives, a recursive algorithm with real-time performance is obtained. In [81], a strategy based on weights to represent the relative importance of several tasks is proposed to deal with transitions while performing a sequence of dynamic tasks with a humanoid robot. In [29], prioritization is treated as a reverse engineering problem based on the hypothesis that the tasks the robot can execute are known, represented in the form of a pool of candidate tasks and associated controllers. This is achieved with the recognition of parallel tasks that have been used to generate a motion, by decoupling the controllers through the nullspace projection operation structure. The error with respect to reference trajectories is thus used to select the best task among a pool of candidates. The effect of this task is then canceled from the original motion by projecting it in the nullspace, where the process continues until the residual motion is null. In [9], the task prioritization problem is treated with a task-parameterized model by exploiting the local linear property of nullspace operators. A set of candidate projections representing multiple task hierarchies is used during demonstration to extract how the different prioritization structures contribute and evolve during the task. This information is then used to generalize the skill to new situations while keeping the same evolution of the prioritization structure as in the demonstrations.

Another category of work focuses on directly learning the nullspace projection operator. In [31], the variability in multiple observations of different tasks is exploited to learn nullspace controllers from demonstrations. In [59], a method is proposed to learn the nullspace projection matrix of a kinematically constrained system by exploiting the property that the nullspace projection can be decomposed into a set of unidimensional projections. This line of work emphasizes that in many everyday behaviors, it is useful to estimate both of the policy underlying the movement and its constraints. In this way, generalization can be achieved both across constraints (i.e., applying the learned policy to new constraints), and within the constraints (i.e., applying new policies to the learned constraint).

Finally, the last category focuses on learning from demonstration, kinesthetic teaching, and interactive refinement techniques that can exploit the nullspace structure of the problem (assumed to be known or acquired by one of the previous approaches). In [83], a method to incrementally learn end-effector and nullspace motions is developed, by kinesthetically teaching nullspace tasks without affecting the end-effector task execution, where a threshold on the external force is used to determine the tasks priority. The resulting controller is able to dynamically switch between multiple tasks priorities, allowing the user to teach the robot motions, consisting of multiple tasks organized in a hierarchical manner, by adding a new task without disturbing the already learned tasks. In [103], a kinesthetic teaching interface is developed to let nonexpert users guide a kinematically redundant robot by its end-effector, with the nullspace structure exploited to let the robot efficiently exploit the redundancy by adopting natural poses while being steered by its end-effector. This allows the user to fully concentrate on the task to achieve, without worrying about the confined spaces or joint limits of the robot.

4.5 Application examples

Examples of applications are shown in Fig. 17. Demonstrations are provided from visual observation of a user executing the task (§2.1). Three different skills are considered: a time-invariant bimanual reaching task, a periodic bimanual sweeping movement, and a physical human-robot interaction (*give-me-five* gesture). Details about the experiments can be found in [12, 86, 56].

In the first two tasks, the control policy is represented by a probabilistic variant of dynamical movement primitives (§3.3) that uses Gaussian mixture regression (§3.4) as core mechanism to learn and generate a virtual mass-spring-damper system with the attractor changing either with the location of the object to track (reaching task) or with time (sweeping motion). The approach also uses multiple coordinate systems acting in parallel, forming a set of candidate frames of reference (of potential relevance for the task). Based on the extracted variations during the observation of the task, the robot determines which of the candidate frame are relevant for the task, and how the different coordinate systems need to be combined to achieve the task (in series and in parallel). In the *give-me-five* interaction task, movements are encoded with continuous hidden Markov models (§3.7), and interaction rules are encoded with discrete hidden Markov models in a hierarchical structure. Based on the symbolic reasoning, the robot trajectories are reshaped in real time.

In the first task, the robot learns that an object on its right-hand side should be reached with the right hand, while the left hand can stay in a comfortable neutral pose, and vice versa if the object is on the left-hand side. It also learns that if the object is at a reachable distance in the middle, both hands can be used to grab the object. In the sweeping task, the robot learns more complex coordinations of the two hands that can keep the broom in contact with the floor and that can adapt to the position and orientation of the area to sweep. In the last task, the robot learns how to acquire movements and physical interaction with a human. From the learned knowledge, it can then recognize human movement, decide about an appropriate interaction strategy, and establish intended physical hand contacts with the human's moving hand.

The approaches in the above experiments can easily be extended to other encoding strategies. For example, although the experiment with COMAN considered a variant of GMM for the encoding, multivariate normal distributions as output, such as hidden Markov models (§3.7), could be used. As shown in [55], hidden Markov models can be combined with Gaussian mixture regression. The experiment used predetermined stiffness and damping parameters, but the retrieved variation and coordination information could easily be exploited in a linear quadratic control strategy (see §4.1 or [15]). Also, the demonstrations were provided so that the relevant features were sufficiently salient to be learned from a small number of demonstrations. To be more efficient, the approach could be augmented with other learning strategies that would allow the humanoid to continue refining the learned skill after the demonstrations, such as leveraging iterative learning control (§4.3) or reward-weighted optimization (§4.2).

5 Future directions and open problems

This chapter introduced several representations for the learning and control of movement primitives in humanoids, together with examples of applications using these techniques. There are many roads for further research, and we introduce in this last section three examples.

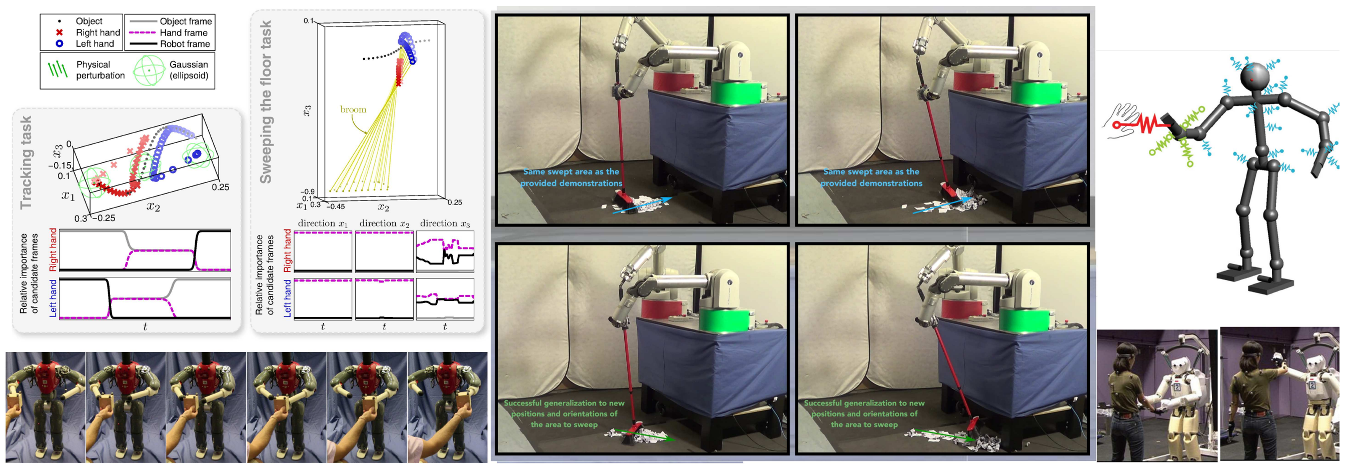


Figure 17: Learning and reproduction of adaptive controllers with the COMAN robot (*left*), with a bimanual platform composed of two Barrett WAMs (*center*), and with the IRT humanoid (*right*). Adapted from [12, 86, 56].

5.1 Learning control representations for wide-ranging data

In the field of machine learning, important efforts are deployed toward developing learning algorithms dedicated to large datasets and deep learning strategies; see, e.g., [58]. Most of these developments target problems in which data are readily available or inexpensive to acquire. Learning control in humanoids holds a distinct challenge, in the sense that it often requires the robot to acquire skills from only few experiences and interactions, with strong generalization demands. On the one side, a humanoid can collect a very large amount of information from a large variety of sensors, but on the other side, it is limited by the number of experiences or demonstrations that the user can provide. Often, such approach requires a simulated environment that can in some cases differ significantly from the behavior of the real platform. Simulators help for one part of the processing, but in some cases, they do not reflect reality in a sufficient level of details to be directly transferred to the real platform without refinement.

For learning control in humanoids, one might endorse the term wide-ranging data (instead of big data), because, on the one hand, several applications in humanoids still require the use of sparse data (and sometimes as a strong requirement, such as learning from demonstration) and, on the other hand, the developed algorithms should be able to exploit further data as efficiently as possible (if available). This challenge is connected to diverse research directions such as online learning, lifelong learning, continual adaptation, or never-ending learning.

5.2 Bridging the gap between symbolic and continuous knowledge

The learning approaches covered in this chapter exploit continuous representations that are tightly linked to the low-level control capability of the humanoids. On the other side of the spectrum, high-level learning approaches exploit discrete representations to provide the level of abstraction required to perform cognitive tasks.

There are research efforts toward augmenting low-level learning methods with the extraction of discrete features and structural elements. Similarly, there are research efforts to provide high-level learning methods with techniques that more closely exploit the motor control capability of the humanoids. Research efforts are required to bridge the gap between symbolic and continuous knowledge in humanoids, which could lead to more flexible and scalable learning of tasks. It requires the development of models and algorithms capable of covering a wide spectrum of representations, from the continuous stream of low-level sensorimotor data to macro actions, reasoning, and higher-level symbolic representations of skills. By starting from the low-level representations discussed in this chapter, one first step in this direction is to investigate the problem of learning to organize in series and in parallel multiple movement primitives (instead of learning each primitive individually), and to tackle the problem of learning the structures of models (instead of setting the structure a priori and learning the parameters).

5.3 Exploiting the social interaction dimension in learning control

Most efforts in learning control have been to develop efficient learning and control algorithms by either assuming that expert datasets are available (e.g., assuming that the provided demonstrations are relevant solutions to the problem) or by predetermining a specific learning strategy, such as:

- Mimicking actions (without understanding the overall objective);
- Goal-level imitation (inverse optimal control, extraction of the underlying objectives by discarding the specific way in which the task is achieved);
- Exploration with self-assessed rewards or feedbacks from an external observer;
- Refinement by kinesthetic corrections.

While such developments are important, they do not account for the way in which data are collected. In contrast to many machine learning applications in which the learning systems are independent of the acquired data, a remarkable characteristic of learning control in humanoids is that the iterative interaction with the users can be exploited to influence the quality and nature of the collected data.

Social learning studies reveal that several modalities need to be combined to acquire skills efficiently [100]. In humanoids, the way in which these different learning control modalities can be organized and coexist remains largely unexplored. Questions include how and when a humanoid should request feedback from the user, either explicitly (e.g., through demonstration requests or spoken questions to validate hypotheses about motor skill properties) or implicitly (e.g., by exaggerating parts of movements to measure users reaction)? How to autonomously determine which learning modality is currently the most appropriate/available/efficient to improve the skill to be acquired? How should this efficiency be measured (e.g., in terms of interaction duration, in terms of generalization ability)? Parts of this problem share links with active learning but with a distinct and important multimodal social interaction aspect.

In addition to extracting control patterns from predetermined learning strategies, one further challenge of learning control in humanoids is to acquire interaction patterns and devise efficient ways of making different learning modalities coexist, such as assessing autonomously which learning strategy to use in a given context. One such research direction requires a better exploitation of the social dimension in human-humanoid interaction, where both actors can influence the success of skills acquisition.

References

[1] Akgun B, Thomaz A (2016) Simultaneously learning actions and goals from demonstration. Autonomous Robots 40(2):211–227

[2] An S, Lee D (2015) Prioritized inverse kinematics with multiple task definitions. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 1423–1430

[3] Anandkumar A, Ge R, Hsu D, Kakade SM, Telgarsky M (2014) Tensor decompositions for learning latent variable models. Journal of Machine Learning Research 15(1):2773–2832

[4] Atkeson CG (1989) Using local models to control movement. In: Advances in Neural Information Processing Systems (NIPS), vol 2, pp 316–323

[5] Atkeson CG, Moore AW, Schaal S (1997) Locally weighted learning for control. Artificial Intelligence Review 11(1-5):75–113

[6] Bristow DA, Tharayil M, Alleyne AG (2006) A survey of iterative learning control. IEEE Control Systems 26(3):96–114

[7] Bryson AE (1999) Dynamic optimization. Addison Wesley Longman Menlo Park

[8] Buchli J, Stulp F, Theodorou E, Schaal S (2011) Learning variable impedance control. Intl Journal of Robotics Research 30(7):820–833

[9] Calinon S (2016) A tutorial on task-parameterized movement learning and retrieval. Intelligent Service Robotics 9(1):1–29, DOI 10.1007/s11370-015-0187-9

[10] Calinon S, Billard AG (2007) Active teaching in robot programming by demonstration. In: Proc. IEEE Intl Symp. on Robot and Human Interactive Communication (Ro-Man), Jeju, Korea, pp 702–707

[11] Calinon S, D'halluin F, Sauser EL, Caldwell DG, Billard AG (2010) Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression. IEEE Robotics and Automation Magazine 17(2):44–54

[12] Calinon S, Li Z, Alizadeh T, Tsagarakis NG, Caldwell DG (2012) Statistical dynamical systems for skills acquisition in humanoids. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Osaka, Japan, pp 323–329

[13] Calinon S, Alizadeh T, Caldwell DG (2013) On improving the extrapolation capability of task-parameterized movement models. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), Tokyo, Japan, pp 610–616

[14] Calinon S, Kormushev P, Caldwell DG (2013) Compliant skills acquisition and multi-optima policy search with EM-based reinforcement learning. Robotics and Autonomous Systems 61(4):369–379

[15] Calinon S, Bruno D, Caldwell DG (2014) A task-parameterized probabilistic model with minimal intervention control. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Hong Kong, China, pp 3339–3344

[16] Cleveland WS (1979) Robust locally weighted regression and smoothing scatterplots. American Statistical Association 74(368):829–836

[17] Dariush B, Gienger M, Jian B, Goerick C, Fujimura K (2008) Whole body humanoid control from human motion descriptors. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2677–2684

[18] Dehio N, Reinhart RF, Steil JJ (2015) Multiple task optimization with a mixture of controllers for motion generation. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), Hamburg, Germany, pp 6416–6421

[19] Demircan E, Sentis L, Sapio VD, Khatib O (2008) Human motion reconstruction by direct control of marker trajectories. In: Advances in Robot Kinematics, pp 263–272

[20] Dietrich A, Albu-Schäffer A, Hirzinger G (2012) On continuous null space projections for torque-based, hierarchical, multi-objective manipulation. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2978–2985

[21] Evrard P, Gribovskaya E, Calinon S, Billard AG, Kheddar A (2009) Teaching physical collaborative tasks: Object-lifting case study with a humanoid. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Paris, France, pp 399–404

[22] Forte D, Gams A, Morimoto J, Ude A (2012) On-line motion synthesis and adaptation using a trajectory database. Robotics and Autonomous Systems 60(10):1327–1339

[23] Furui S (1986) Speaker-independent isolated word recognition using dynamic features of speech spectrum. IEEE Trans on Acoustics, Speech, and Signal Processing 34(1):52–59

[24] Gams A, Van den Kieboom J, Vespignani M, Guyot L, Ude A, Ijspeert A (2014) Rich periodic motor skills on humanoid robots: Riding the pedal racer. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2326–2332

[25] Gams A, Nemec B, Ijspeert A, Ude A (2014) Coupling movement primitives: Interaction with the environment and bimanual tasks. IEEE Trans on Robotics 30(4):816–830

[26] Giese MA, Mukovskiy A, Park AN, Omlor L, Slotine JJE (2009) Real-time synthesis of body movements based on learned primitives. In: Statistical and Geometrical Approaches to Visual Motion Analysis: Intl Dagstuhl Seminar, Springer-Verlag, Berlin, Heidelberg, pp 107–127

[27] Gómez V, Kappen HJ, Peters J, Neumann G (2014) Policy search for path integral control. In: Proc. European Conf. on Machine Learning and Knowledge Discovery in Databases, Springer-Verlag New York, Inc., New York, NY, USA, vol 8724, pp 482–497

[28] Gonzalez-Fierro M, Balaguer C, Swann N, Nanayakkara T (2014) Full-body postural control of a humanoid robot with both imitation learning and skill innovation. International Journal of Humanoid Robotics 11(2):1–34

[29] Hak S, Mansard N, Stasse O, Laumond JP (2012) Reverse control for humanoid robot task recognition. IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics 42(6):1524–1537

[30] Hersch M, Guenter F, Calinon S, Billard AG (2008) Dynamical system modulation for robot learning via kinesthetic demonstrations. IEEE Trans on Robotics 24(6):1463–1467

[31] Howard M, Klanke S, Gienger M, Goerick C, Vijayakumar S (2008) Behaviour generation in humanoids by learning potential-based policies from constrained motion. Appl Bionics Biomechanics 5(4):195–211

[32] Hu K, Lee D (2012) Prediction-based synchronized human walking motion imitation by a humanoid robot. Automatisierungstechnik 60 (11):705–714

[33] Hu K, Ott C, Lee D (2014) Online human walking imitation in task and joint space based on quadratic programming. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), IEEE, pp 3458–3464

[34] Hu K, Ott C, Lee D (2015) Online iterative learning control of zero-moment point for biped walking stabilization. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 5127–5133

[35] Hwangbo J, Gehring C, Sommer H, Siegwart R, Buchli J (2014) ROCK*: Efficient black-box optimization for policy learning. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), pp 535–540

[36] Ijspeert A, Nakanishi J, Pastor P, Hoffmann H, Schaal S (2013) Dynamical movement primitives: Learning attractor models for motor behaviors. Neural Computation 25(2):328–373

[37] Ijspeert AJ, Nakanishi J, Schaal S (2001) Trajectory formation for imitation with nonlinear dynamical systems. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pp 752–757

[38] Inamura T, Kojo N, Inaba M (2006) Situation recognition and behavior induction based on geometric symbol representation of multimodal sensorimotor patterns. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pp 5147–5152

[39] Kern S, Mueller SD, Hansen N, Bueche D, Ocenasek J, Koumoutsakos P (2004) Learning probability distributions in continuous evolutionary algorithms - a comparative review. Natural Computing 3(1):77–112

[40] Khansari-Zadeh SM, Billard A (2011) Learning stable nonlinear dynamical systems with Gaussian mixture models. IEEE Trans on Robotics 27(5):943–957

[41] Khansari-Zadeh SM, Billard A (2014) Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. Robotics and Autonomous Systems 62(6):752–765

[42] Kim S, Kim C, Park JH (2006) Human-like arm motion generation for humanoid robots using motion capture database. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pp 3486–3491

[43] Kim S, Hong S, Kim D (2010) A walking motion imitation framework of a humanoid robot by human walking recognition from IMU motion data. In: 9th IEEE-RAS International Conference on Humanoid Robots, pp 343–348

[44] Kim S, Shukla A, Billard A (2014) Catching objects in flight. IEEE Trans on Robotics 30(5):1049–1065

[45] Kober J, Peters J (2010) Imitation and reinforcement learning: Practical algorithms for motor primitives in robotics. IEEE Robotics and Automation Magazine 17(2):55–62

[46] Koenemann J, Burget F, Bennewitz M (2014) Real-time imitation of human whole-body motions by humanoids. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2806–2812

[47] Kormushev P, Calinon S, Saegusa R, Metta G (2010) Learning the skill of archery by a humanoid robot iCub. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Nashville, TN, USA, pp 417–423

[48] Kormushev P, Nenchev DN, Calinon S, Caldwell DG (2011) Upper-body kinesthetic teaching of a free-standing humanoid robot. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Shanghai, China, pp 3970–3975

[49] Kormushev P, Ugurlu B, Calinon S, Tsagarakis N, Caldwell DG (2011) Bipedal walking energy minimization by reinforcement learning with evolving policy parameterization. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), San Francisco, CA, USA, pp 318–324

[50] Kroese DP, Porotsky S, Rubinstein RY (2006) The cross-entropy method for continuous multi-extremal optimization. Methodology and Computing in Applied Probability 8:383–407

[51] Kulic D, Ott C, Lee D, Ishikawa J, Nakamura Y (2012) Incremental learning of full body motion primitives and their sequencing through human motion observation. Intl Journal of Robotics Research 31(3):330–345

[52] Kwon J, Park FC (2008) Natural movement generation using hidden Markov models and principal components. IEEE Trans on Systems, Man, and Cybernetics, Part B 38(5):1184–1194

[53] de Lasa M, Hertzmann A (2009) Prioritized optimization for task-space control. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), St Louis, MO, USA, pp 5755–5762

[54] Lee D, Ott C (2010) Incremental motion primitive learning by physical coaching using impedance control. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pp 4133–4140

[55] Lee D, Ott C (2011) Incremental kinesthetic teaching of motion primitives using the motion refinement tube. Autonomous Robots 31(2):115–131

[56] Lee D, Ott C, Nakamura Y (2010) Mimetic communication model with compliant physical contact in human-humanoid interaction. Intl Journal of Robotics Research 29(13):1684–1704

[57] Lee SH, Suh IH, Calinon S, Johansson R (2012) Learning basis skills by autonomous segmentation of humanoid motion trajectories. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Osaka, Japan, pp 112–119

[58] Levine S, Finn C, Darrell T, Abbeel P (2016) End-to-end training of deep visuomotor policies. Journal of Machine Learning Research 17(39):1–40

[59] Lin HC, Howard M, Vijayakumar S (2015) Learning null space projections. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2613–2619

[60] Liu M, Tan Y, Padois V (2016) Generalized hierarchical control. Autonomous Robots 40(1):17–31

[61] Lober R, Padois V, Sigaud O (2014) Multiple task optimization using dynamical movement primitives for whole-body reactive control. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Madrid, Spain, pp 193–198

[62] Lober R, Padois V, Sigaud O (2015) Variance modulated task prioritization in whole-body control. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), Hamburg, Germany, pp 3944–3949

[63] Longman RW, Mombaur KD (2006) Investigating the use of iterative learning control and repetitive control to implement periodic gaits. In: Fast motions in biomechanics and robotics, Springer, pp 189–218

[64] Maeda GJ, Neumann G, Ewerton M, Lioutikov R, Kroemer O, Peters J (2017) Probabilistic movement primitives for coordination of multiple human-robot collaborative tasks. Autonomous Robots 41(3):593–612, DOI 10.1007/s10514-016-9556-2

[65] Medina JR, Lee D, Hirche S (2012) Risk-sensitive optimal feedback control for haptic assistance. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 1025–1031

[66] Modugno V, Neumann G, Rueckert E, Oriolo G, Peters J, Ivaldi S (2016) Learning soft task priorities for control of redundant robots. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Stockholm, Sweden

[67] Moro FL, Gienger M, Goswami A, Tsagarakis NG (2013) An attractor-based whole-body motion control (WBMC) system for humanoid robots. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Atlanta, GA, USA, pp 42–49

[68] Mühlig M, Gienger M, Steil J (2012) Interactive imitation learning of object movement skills. Autonomous Robots 32(2):97–114

[69] Nakanishi J, Morimoto J, Endo G, Cheng G, Schaal S, Kawato M (2004) Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems 47(2-3):79–91

[70] Neumann K, Steil JJ (2015) Learning robot motions with stable dynamical systems under diffeomorphic transformations. Robot Auton Syst 70:1–15

[71] Ott C, Lee D, Nakamura Y (2008) Motion capture based human motion recognition and imitation by direct marker control. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), pp 399–405

[72] Ott C, Henze B, Lee D (2013) Kinesthetic teaching of humanoid motion based on whole-body compliance control with interaction-aware balancing. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pp 4615–4621

[73] Paraschos A, Daniel C, Peters J, Neumann G (2013) Probabilistic movement primitives. In: Advances in Neural Information Processing Systems (NIPS), Curran Associates, Inc., pp 2616–2624

[74] Perrin N, Schlehuber-Caissier P (2016) Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. Systems & Control Letters 96:51–59

[75] Peters RA, Campbell CL, Bluethmann WJ, Huber E (2003) Robonaut task learning through teleoperation. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2806–2811

[76] Pollard N, Hodgins J, Riley M, Atkeson C (2002) Adapting human motion for the control of a humanoid robot. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 1390–1397

[77] Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77:2:257–285

[78] Roberts S, Osborne M, Ebden M, Reece S, Gibson N, Aigrain S (2012) Gaussian processes for time-series modelling. Philosophical Trans of the Royal Society A 371(1984):1–25

[79] Rozo L, Silvério J, Calinon S, Caldwell DG (2016) Learning controllers for reactive and proactive behaviors in human-robot collaboration. Frontiers in Robotics and AI 3(30):1–11, DOI 10.3389/frobt.2016.00030

[80] Rueckert E, Mundo J, Paraschos A, Peters J, Neumann G (2015) Extracting low-dimensional control variables for movement primitives. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Seattle, WA, USA, pp 1511–1518

[81] Salini J, Padois V, Bidaud P (2011) Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 1283–1290

[82] Sauser EL, Argall BD, Metta G, Billard AG (2012) Iterative learning of grasp adaptation through human corrections. Robot Auton Syst 60(1):55–71

[83] Saveriano M, An S, Lee D (2015) Incremental kinesthetic teaching of end-effector and null-space motion primitives. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 3570–3575

[84] Schaal S, Atkeson CG (1998) Constructive incremental learning from only local information. Neural Computation 10(8):2047–2084

[85] Schreiter J, Englert P, Nguyen-Tuong D, Toussaint M (2015) Sparse Gaussian process regression for compliant, real-time robot control. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), pp 2586–2591

[86] Silvério J, Rozo L, Calinon S, Caldwell DG (2015) Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), Hamburg, Germany, pp 464–470

[87] Stulp F, Sigaud O (2012) Path integral policy improvement with covariance matrix adaptation. In: Proc. Intl Conf. on Machine Learning (ICML), pp 1–8

[88] Stulp F, Sigaud O (2013) Robot skill learning: From reinforcement learning to evolution strategies. Paladyn Journal of Behavioral Robotics 4(1):49–61

[89] Stulp F, Sigaud O (2015) Many regression algorithms, one unified model - a review. Neural Networks 69:60–79

[90] Stulp F, Buchli J, Theodorou E, Schaal S (2010) Reinforcement learning of full-body humanoid motor skills. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), Nashville, TN, USA, pp 405–410

[91] Sugimoto N, Morimoto J (2013) Trajectory-model-based reinforcement learning: Application to bimanual humanoid motor learning with a closed-chain constraint. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids), pp 429–434

[92] Sugiura K, Iwahashi N, Kashioka H, Nakamura S (2011) Learning, generation, and recognition of motions by reference-point-dependent probabilistic models. Advanced Robotics 25(6-7):825–848

[93] Takano W, Nakamura Y (2016) Real-time unsupervised segmentation of human whole-body motion and its application to humanoid robot acquisition of motion symbols. Robotics and Autonomous Systems 75, Part B:260–272

[94] Tanwani AK, Calinon S (2016) Learning robot manipulation tasks with task-parameterized semi-tied hidden semi-Markov model. IEEE Robotics and Automation Letters (RA-L) 1(1):235–242, DOI 10.1109/LRA.2016.2517825

[95] Ting J, Kalakrishnan M, Vijayakumar S, Schaal S (2008) Bayesian kernel shaping for learning control. In: Advances in Neural Information Processing Systems (NIPS), pp 1673–1680

[96] Todorov E, Jordan MI (2002) Optimal feedback control as a theory of motor coordination. Nature Neuroscience 5:1226–1235

[97] Ude A, Gams A, Asfour T, Morimoto J (2010) Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans on Robotics 26(5):800–815

[98] Vijayakumar S, D'souza A, Schaal S (2005) Incremental online learning in high dimensions. Neural Computation 17(12):2602–2634

[99] Werner A, Trautmann D, Lee D, Lampariello R (2015) Generalization of optimal motion trajectories for bipedal walking. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), pp 1571–1577

[100] Whiten A, McGuigan N, Marshall-Pescini S, Hopper LM (2009) Emulation, imitation, over-imitation and the scope of culture for child and chimpanzee. Phil Trans R Soc B 364(1528):2417–2428

[101] Williams CKI, Rasmussen CE (1996) Gaussian processes for regression. In: Advances in Neural Information Processing Systems (NIPS), pp 514–520

[102] Wilson AG, Ghahramani Z (2011) Generalised Wishart processes. In: Annual Conf. on Uncertainty in Artificial Intelligence, Barcelona, Spain

[103] Wrede S, Emmerich C, Ricarda R, Nordmann A, Swadzba A, Steil JJ (2013) A user study on kinesthetic teaching of redundant robots in task and configuration space. Journal of Human-Robot Interaction 2(1):56–81

[104] Zeestraten MJA, Calinon S, Caldwell DG (2016) Variable duration movement encoding with minimal intervention control. In: Proc. IEEE Intl Conf. on Robotics and Automation (ICRA), Stockholm, Sweden, pp 497–503