A Tutorial on Task-Parameterized Movement Learning and Retrieval

Sylvain Calinon*

Abstract

Task-parameterized models of movements aim at automatically adapting movements to new situations encountered by a robot. The task parameters can for example take the form of positions of objects in the environment, or landmark points that the robot should pass through. This tutorial aims at reviewing existing approaches for task-adaptive motion encoding. It then narrows down the scope to the special case of task parameters that take the form of frames of reference, coordinate systems, or basis functions, which are most commonly encountered in service robotics. Each section of the paper is accompanied with source codes designed as simple didactic examples implemented in *Matlab* with a full compatibility with *GNU Octave*, closely following the notation and equations of the article. It also presents ongoing work and further challenges that remain to be addressed, with examples provided in simulation and on a real robot (transfer of manipulation behaviors to the Baxter bimanual robot). The repository for the accompanying source codes is available at `http://www.idiap.ch/software/pbdlib/`.

Keywords: Probabilistic motion encoding, Task-parameterized movements, Task-adaptive models, Natural motion synthesis

1 Introduction

In contrast to industrial robots in large factories, a wide range of service robots are designed to move in unconstrained environments in which they should fulfill a series of tasks while swiftly reacting to perturbations. The expectations and promises of service robotics applications are very challenging and cannot be achieved without joint efforts from different fields of robotics. This exploitation of various methods can hardly be done serially, and instead requires closer interactions between learning, planning and control. One of the prior requirement to face such challenge is to design a versatile representation of what the robot should do (how it should move, which behavior it should follow) that is compatible with the above techniques and that can be shared bilaterally. In particular, in continuously changing environments, the movements of service robots need to be generated and adapted to the ongoing situation very quickly.

This tutorial takes the perspective that the challenges of recognizing, predicting and generating movements can be achieved within the same encoding strategy. It will show that simple probabilistic mixture models can be exploited to model the natural variations in human and robot motions, as well as to make links between learning, online planning and optimal control. Gaussian mixture models provide a structure that is compatible with many robot learning approaches. It is flexible to the requirements of service robotics, because the representation can be easily adapted to the application requirements while preserving the core probabilistic mixture modeling strategy (addition of transition information in the form of an HMM, subspace clustering with MFA or MPPCA, etc.). Finally, the model is not tied to a specific parameters estimation technique, which allows the movements to be acquired by different interaction modalities and learning strategies.

The tutorial will focus on *task-parameterized Gaussian mixture model* (TP-GMM), by presenting a number of extensions and ongoing challenges targeting applications in unconstrained environment.

1.1 Organization of the paper

Section 2 discusses the importance of considering adaptive models of movements in robotics. It introduces the proposed approach from a high-level perspective and motivates it by using a toy example with a single Gaussian.

Section 3 presents the core of the approach by taking the example of a standard *Gaussian mixture model* (GMM) modified as a *task-parameterized GMM* (TP-GMM). It also discusses the practical use of regularization terms.

The next four sections present techniques that rely on the core TP-GMM encoding strategy but that tackle different challenges, by moving progressively from encoding issues to kinematic and dynamic retrieval of data. Section 4 addresses the challenge of handling high dimensional data with subspace clustering extensions of the model. Section 5 discusses the challenge of generating continuous movements from task-parameterized models. It presents two distinct approaches based on Gaussian mixture regression (Section 5.1) or trajectory models encoding of dynamic features (Section 5.2).

Section 6 then extends the motion synthesis challenge to the important problem of learning a controller for the robot, by presenting a minimal intervention control strategy that can exploit the proposed task parameterized model.

Section 7 gives an overview of more advanced forms of task parameterization that can be considered, such as constraints in different data spaces (e.g., to handle constraints at joint and end-effector levels simultaneously), as well as priority and projection constraints.

Finally, Section 8 presents comparisons with other task-adaptive approaches, and Section 9 discusses further work.

Each section of the paper is accompanied with source codes designed as simple didactic examples implemented

*Idiap Research Institute, Martigny, Switzerland, E-mail: `sylvain.calinon@idiap.ch`. This work was in part supported by the DexROV Project through the EC Horizon 2020 programme (Grant #635491). The final publication is available at Springer via `http://dx.doi.org/10.1007/s11370-015-0187-9`.

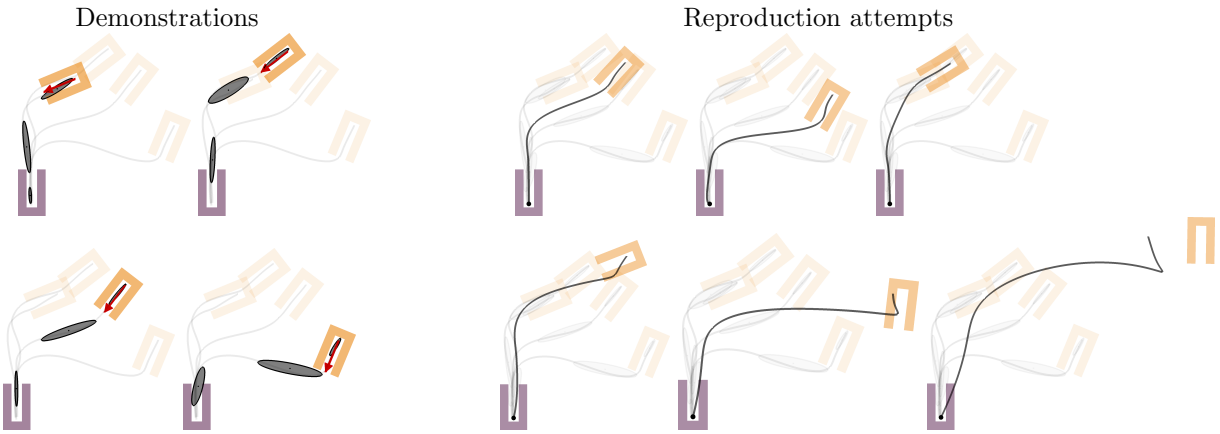


Figure 1: Solving a task adaptation problem as a standard regression problem does not always provide satisfying generalization capability. *Left:* 4 demonstrations where the task parameters are treated as inputs (position and direction of the second object concatenated in a vector), and where the motion model parameters are treated as outputs (GMM parameters concatenated in a vector). Here, each demonstration is aligned by fitting a single model to the concatenated set of demonstrations. *Right:* Reproduction attempts by treating the problem of adapting the movement to the new task parameters as standard regression, namely by relying on the training set to regenerate new model parameters based on new task parameters, and by using this model to generate a new motion with Gaussian mixture regression. Here, both inputs and outputs are treated as multidimensional vectors. The 6 reproduction attempts consider situations of increasing complexity. We can observe that the system provides good interpolation results but cannot extrapolate well when faced with situations that are far from the regions covered by the demonstrations. This example was implemented with Gaussian process regression, but similar reproduction results are observed with other regression mechanisms. As expected, reproductions far outside the regions covered by the demonstrations will tend to collapse to an average of the different trajectory models, resulting in poor generalization capability.

in *Matlab/GNU Octave*. In order to facilitate reading, implementation details such as estimation update rules have been gathered at the end of the paper in the form of Appendices.

2 Adaptive models of movements

Task-parameterized models of movements/behaviors refer to representations that can automatically adapt to a set of task parameters that can, for example, describe the current context, situation, state of the environment, or state of the robot configuration. The *task parameters* refer to the variables that can be collected by the system and that describe a situation, such as positions of objects in the environment. The *task parameters* can be fixed during an execution trial or they can vary while the motion is executed. The *model parameters* refer to the variables learned by the system, namely, that are stored in memory (the internal representation of the movement). During reproduction, a new set of *task parameters* (description of the present situation) is used to produce new movements (e.g., adaptation to new position of objects after having observed the skill in a different situation).

Several denominations have been introduced in the literature to describe these models, such as *task-parameterized* [91, 63, 20] (the denomination used here), *parametric* [102, 51, 59], *stylistic* [13] or *object-centric warping* [56]. In these models, the encoding of skills usually serve several purposes, including classification, prediction, synthesis and online adaptation. A taxonomy of task-parameterized models is presented in [14], with three broad categories, namely:

1. Approaches employing M models for the M demonstrations, performed in M different situations, see e.g. [29, 47, 59, 50, 97, 21, 44];

2. Approaches employing P models for the P frames of reference that are possibly relevant for the task, see e.g. [3, 65, 25];

3. Approaches employing a single model whose parameters are modulated by task parameters, see e.g. [102, 51, 43, 79, 71, 70].

In the majority of these approaches, the retrieval of movements from the model parameters and the task parameters is viewed as a regression problem. This generality might look appealing at first sight, but it also limits the generalization scope of these models, see Fig. 1. Task-parameterized Gaussian mixture models (TP-GMM) aims at increasing this generalization capability by exploiting the functional nature of task parameters. Indeed, in robotics applications, task parameters can most of the time be related to frames of reference, coordinate systems, basis functions or local projections, whose structure can be exploited to speed up learning and provide the system with better extrapolation capability.

2.1 Proposed approach

The proposed approach uses a generative model to encode the movement, where the variability and correlation information is used to infer the impedance parameters of a virtual spring-damper system. These parameters figuratively correspond to the stiffness of a spring and to the damping

Table 1: Notation and names of variables.

Dimensions:	
T	Number of datapoints in a trajectory (t will be used as index)
N	Number of datapoints in a training set (t will be used as index)
M	Number of demonstrated trajectories in a training set (m will be used as index)
D	Dimension of a datapoint
C	Number of derivatives (including position) to represent the state space ($C=2$ for $[\boldsymbol{x}^\top, \dot{\boldsymbol{x}}^\top]^\top$)
d	Dimension of the subspace in which datapoints are projected
K	Number of Gaussian components in a mixture model (i and k will be used as indices)
P	Number of candidate frames in a task-parameterized mixture (j will be used as index/exponent)
Distributions:	
$\mathcal{N}(\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)})$	i-th multivariate Gaussian distribution in frame j of a TP-GMM
$\boldsymbol{\mu}_i^{(j)}$	Center of the Gaussian
$\boldsymbol{\Sigma}_i^{(j)}$	Covariance matrix ($\sigma_i^{2(j)}$ for unidimensional Gaussian)
π_i	Prior probability
$\mathcal{P}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{O}})$,	Joint probability of $\boldsymbol{x}^{\mathcal{I}}$ and $\boldsymbol{x}^{\mathcal{O}}$
$\mathcal{P}(\boldsymbol{x}^{\mathcal{O}}\|\boldsymbol{x}^{\mathcal{I}})$,	Conditional probability of $\boldsymbol{x}^{\mathcal{O}}$ given $\boldsymbol{x}^{\mathcal{I}}$
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Likelihood of $\boldsymbol{x}$ to be sampled from the normal distribution with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$
Feature spaces:	
$\boldsymbol{x}$	Position in Cartesian space (operational space)
$\boldsymbol{u}$	Control command in operational space (acceleration in Cartesian space)
$\boldsymbol{q}$	Position in joint space (configuration space)
$\boldsymbol{\xi}$	Used to describe a generic multidimensional vector or matrix (e.g., $\boldsymbol{\xi} = [\boldsymbol{x}^\top, \dot{\boldsymbol{x}}^\top]^\top$)
$\boldsymbol{A}_{t,j}$	Linear transformation matrix describing frame j at time step t (e.g., orientation of object j)
$\boldsymbol{b}_{t,j}$	Offset vector describing frame j at time step t (e.g., location of object j)
Linear algebra operators:	
$\boldsymbol{\xi}^\top$	Transpose of matrix/vector $\boldsymbol{\xi}$
$\boldsymbol{\xi}^{-1}$	Inverse of a square matrix $\boldsymbol{\xi}$
$\boldsymbol{\xi}^\dagger$	Pseudoinverse of matrix/vector $\boldsymbol{\xi}$
$\dot{\boldsymbol{\xi}}$	Velocity (for corresponding position $\boldsymbol{\xi}$)
$\ddot{\boldsymbol{\xi}}$	Acceleration (for corresponding position $\boldsymbol{\xi}$)
$\boldsymbol{\xi}^{\mathcal{I}}$	Subset of a multidimensional vector/matrix that spans input dimensions
$\boldsymbol{\xi}^{\mathcal{O}}$	Subset of a multidimensional vector/matrix that spans output dimensions
$\boldsymbol{0}$	Matrix with all elements being zeros
$\boldsymbol{I}$	Identity matrix

Table 2: List of Matlab/GNU Octave examples (by alphabetic order). The repository for the accompanying source codes is available at `http://www.idiap.ch/software/pbdlib/`.

Filename	**Description**
benchmark_DS_GP_GMM01	Benchmark of task-parameterized model based on Gaussian process regression, with trajectory model (Gaussian mixture model encoding)
benchmark_DS_GP_raw01	Same as benchmark_DS_GP_GMM01 but with raw trajectory
benchmark_DS_PGMM01	Benchmark of task-parameterized model based on parametric Gaussian mixture model (PGMM)
benchmark_DS_TP_GMM01	Benchmark of task-parameterized Gaussian mixture model (TP-GMM)
benchmark_DS_TP_GP01	Benchmark of task-parameterized Gaussian process (nonparametric task-parameterized method)
benchmark_DS_TP_LWR01	Benchmark of task-parameterized locally weighted regression (nonparametric task-parameterized method)
benchmark_DS_TP_MFA01	Benchmark of task-parameterized mixture of factor analyzers (TP-MFA)
benchmark_DS_TP_trajGMM01	Benchmark of task-parameterized trajectory-GMM
demo_affineTransform01	Affine transformations of raw data as pre-processing step to train a task-parameterized model
demo_batchLQR01	Controller retrieval through a batch solution of linear quadratic optimal control (unconstrained linear MPC), by relying on a Gaussian mixture model (GMM) encoding of position and velocity data (see also demo_iterativeLQR01)
demo_batchLQR02	Same as demo_batchLQR01 but with only position data
demo_DMP_GMR01	Emulation of a standard dynamic movement primitive (DMP) by using a GMM with diagonal covariance matrix, and retrieval computed through Gaussian mixture regression (GMR)
demo_DMP_GMR02	Same as demo_DMP_GMR01 but with full covariance matrices coordinating the different variables
demo_DMP_GMR03	Same as demo_DMP_GMR02 but with GMR used to regenerate the path of a spring-damper system instead of encoding the nonlinear forcing term
demo_DMP_GMR04	Same as demo_DMP_GMR03 by using the task-parameterized model formalism
demo_DMP_GMR_LQR01	Same as demo_DMP_GMR04 but with LQR used to refine the parameters of the spring-damper system
demo_DMP_GMR_LQR02	Same as demo_DMP_GMR_LQR01 with perturbations added to show the benefit of full covariance to coordinate disturbance rejection
demo_DSGMR01	Gaussian mixture model (GMM), with a dynamical system based on Gaussian mixture regression(GMR) driven by a decay term (as in DMP)
demo_DTW01	Trajectory realignment through dynamic time warping (DTW)
demo_GMM01	Gaussian mixture model (GMM) parameters estimation
demo_GMR01	GMM with time-based Gaussian mixture regression (GMR) used for reproduction
demo_GPR01	Use of Gaussian process regression (GPR) as a task-parameterized model
demo_HDDC01	High Dimensional Data Clustering model (HDDC, HD-GMM)
demo_iterativeLQR01	Controller retrieval through an iterative solution of linear quadratic optimal control (finite horizon, unconstrained linear MPC), by relying on a GMM encoding of position and velocity data (see also demo_batchLQR01)
demo_iterativeLQR02	Same as demo_iterativeLQR01 with only position data
demo_MFA01	Mixture of factor analysers (MFA) parameters estimation
demo_MPPCA01	Mixture of probabilistic principal component analyzers (MPPCA) parameters estimation
demo_stdPGMM01	Parametric Gaussian mixture model (PGMM) used as a task-parameterized model, with DS-GMR employed to retrieve continuous movements
demo_testDampingRatio01	Test with critically damped system and ideal underdamped system
demo_testLQR01	Test of linear quadratic regulation (LQR) with different variance in the data
demo_testLQR02	Test of LQR with evaluation of the damping ratio found by the system
demo_testLQR03	Comparison of LQR with finite and infinite time horizons
demo_testLQR04	Demonstration of the coordination capability of linear quadratic optimal control when combined with full precision matrices
demo_TPbatchLQR01	Task-parameterized GMM encoding position and velocity data, combined with a batch solution of linear quadratic optimal control
demo_TPbatchLQR02	Batch solution of a linear quadratic optimal control acting in multiple frames, which is equivalent to TP-GMM combined with LQR
demo_TPGMM01	Task-parameterized Gaussian mixture model (TP-GMM) encoding
demo_TPGMR01	TP-GMM with GMR used for reproduction (without dynamical system)
demo_TPGMR_DS01	Dynamical system with constant gains used with a task-parameterized model
demo_TPGMR_LQR01	Finite horizon LQR used with a task-parameterized model
demo_TPGMR_LQR02	Infinite horizon LQR used with a task-parameterized model
demo_TPGP01	Task-parameterized Gaussian process regression (TP-GPR)
demo_TPHDDC01	Task-parameterized high dimensional data clustering (TP-HDDC)
demo_TPMFA01	Task-parameterized mixture of factor analyzers (TP-MFA)
demo_TPMPC01	Task-parameterized model encoding position data, with MPC used to track the associated stepwise reference path
demo_TPMPC02	Same as demo_TPMPC01 with a generalized version of MPC used to track associated stepwise reference paths in multiple frames
demo_TPMPPCA01	Task-parameterized mixture of probabilistic principal component analyzers (TP-MPPCA)
demo_TPtrajGMM01	Task-parameterized model with trajectory-GMM encoding
demo_trajGMM01	Reproduction of trajectory with a GMM with dynamic features (trajectory-GMM)
demo_trajMFA01	Trajectory model with either a mixture of factor analysers (MFA), a mixture of probabilistic principal component analyzers (MPPCA), or a high-dimensional data clustering approach (HD-GMM)
demoIK_nullspace_TPGMM01	Inverse kinematics with nullspace treated with task-parameterized GMM (bimanual tracking task, version with 4 frames)
demoIK_pointing_TPGMM01	Task-parameterized GMM to encode pointing direction by considering nullspace constraint (4 frames) (example with two objects and robot frame, starting from the same initial pose (nullspace constraint), by using a single Euler orientation angle and 3 DOFs robot)

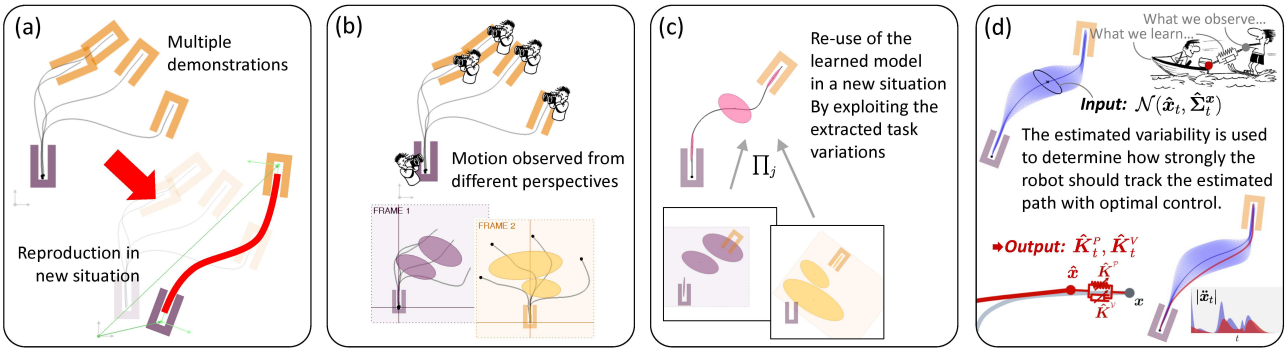


Figure 2: Illustration of the overall approach (see main text for details). *(a)* Observation of a task in different situations and generalization to new contexts. Multiple demonstrations provide the opportunity to discern the structure of the task. *(b)* Probabilistic encoding of continuous movements in multiple coordinate systems. *(c)* Exploitation of variability and correlation information to adapt the motion to new situations. With cross-situational observations of the same task, the robot can generalize the skill to new situations. *(d)* Computation of the underlying optimal control strategy driving the observed behavior.

coefficient of a viscous damper, with the difference that they can also be full stiffness and damping matrices.

In its task-parameterized version, the model uses several frames of reference to describe the robot behavior in multiple coordinate systems. The variations and correlations observed from the perspective of these different frames are exploited to determine the impedance of the system with a linear quadratic regulator. Fig. 2 illustrates the overall approach, which can be decomposed into multiple steps, involving statistical modeling, dynamical systems and optimal control. This illustration will be used as a guiding thread to describe throughout the article the different model components and algorithms enabling *learning, adaptation, synthesis and control* of movement skills.

The proposed task-parameterized model is not new: preliminary versions were investigated in [20, 14, 16] for the special case of frames of reference representing rotations and translations in Cartesian space. The current paper discusses the potentials of the approach, and introduces several routes for further investigation, which aim at applying the proposed technique to a wider range of affine transformations (directly exploiting the robotics application domain), including constraints in both configuration and operational spaces, as well as priority constraints. It also shows that the proposed method can be applied to different probabilistic encoding strategies, including subspace clustering approaches that enable the model to handle feature spaces of high dimensions.

Table 1 will be used as a reference to the notation, dimensions and names of variables employed in the paper. As a general rule, lowercase and uppercase bold fonts respectively indicate vectors and matrices, while normal fonts indicate scalars. Table 2 lists all examples available as *Matlab/GNU Octave* source codes.

2.2 Example with a single Gaussian

Before presenting the details of the task-parameterized model, the approach is motivated by an introductory example with a single Gaussian. Two frames are considered, described respectively at each time step t by $\{\boldsymbol{b}_{t,1}, \boldsymbol{A}_{t,1}\}$ and $\{\boldsymbol{b}_{t,2}, \boldsymbol{A}_{t,2}\}$, representing the origin of the observer $\boldsymbol{b}$ and a set of basis vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots\}$ forming a transformation matrix $\boldsymbol{A} = [\boldsymbol{e}_1 \boldsymbol{e}_2 \cdots]$.

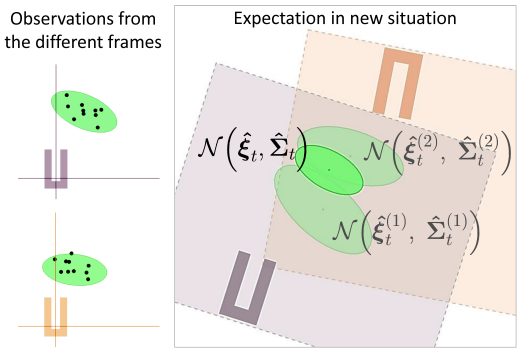


Figure 3: Minimization of the objective function in Eq. (3) composed of a weighted sum of quadratic error terms, whose result corresponds to a product of Gaussians.

A set of demonstrations is observed from the perspective of the two frames. During reproduction, each frame expects the new datapoints to lie within the same range. If $\mathcal{N}\left(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}\right)$ and $\mathcal{N}\left(\boldsymbol{\mu}^{(2)}, \boldsymbol{\Sigma}^{(2)}\right)$ describe the observations in the first and second frames, the two observers respectively expect the reproduction attempts to lie within the distributions $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_t^{(1)}, \hat{\boldsymbol{\Sigma}}_t^{(1)}\right)$ and $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_t^{(2)}, \hat{\boldsymbol{\Sigma}}_t^{(2)}\right)$ with

$$\hat{\boldsymbol{\xi}}_t^{(1)} = \boldsymbol{A}_{t,1}\,\boldsymbol{\mu}^{(1)} + \boldsymbol{b}_{t,1} \quad , \quad \hat{\boldsymbol{\Sigma}}_t^{(1)} = \boldsymbol{A}_{t,1}\,\boldsymbol{\Sigma}^{(1)}\,\boldsymbol{A}_{t,1}^{\top}, \qquad (1)$$

$$\hat{\boldsymbol{\xi}}_t^{(2)} = \boldsymbol{A}_{t,2}\,\boldsymbol{\mu}^{(2)} + \boldsymbol{b}_{t,2} \quad , \quad \hat{\boldsymbol{\Sigma}}_t^{(2)} = \boldsymbol{A}_{t,2}\,\boldsymbol{\Sigma}^{(2)}\,\boldsymbol{A}_{t,2}^{\top}, \qquad (2)$$

computed with the linear transformation property of normal distributions.

During reproduction, a trade-off needs to be determined to concord with the distributions expected by each frame. The underlying objective function is defined as the weighted sum of the quadratic error terms

$$\hat{\boldsymbol{\xi}}_t = \arg\min_{\boldsymbol{\xi}_t} \sum_{j=1}^{2} \left(\boldsymbol{\xi}_t - \hat{\boldsymbol{\xi}}_t^{(j)}\right)^{\top} \hat{\boldsymbol{\Sigma}}_t^{(j)\,-1} \left(\boldsymbol{\xi}_t - \hat{\boldsymbol{\xi}}_t^{(j)}\right). \qquad (3)$$

The above objective can be solved easily by differentiating and equating to zero the above equation, yielding a point $\hat{\boldsymbol{\xi}}_t$, with an estimation error defined by a covariance $\hat{\boldsymbol{\Sigma}}_t$. It is easy to show that the resulting $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_t, \hat{\boldsymbol{\Sigma}}_t\right)$ corresponds to the product of the two Gaussians $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_t^{(1)},\ \hat{\boldsymbol{\Sigma}}_t^{(1)}\right)$ and $\mathcal{N}\left(\hat{\boldsymbol{\xi}}_t^{(2)},\ \hat{\boldsymbol{\Sigma}}_t^{(2)}\right)$, see [18] for details.

Fig. 3 illustrates this process for one of the Gaussian in Fig. 2.

3 Task-parameterized Gaussian mixture model (TP-GMM)

The *task-parameterized Gaussian mixture model* (TP-GMM) is a direct extension of the objective problem presented above, by considering multiple frames and multiple clusters of datapoints (soft clustering via mixture modeling). It probabilistically encodes the relevance of candidate frames, which can change during the task. In contrast to approaches such as [70] that aim at extracting a single (most prominent) coordinate system located at the end of a motion segment, the proposed approach allows the superposition and transition of different coordinate systems that are relevant for the task (parallel organization of behavior primitives, adaptation to multiple viapoints in the middle of the movement, or modulation based on positions, orientations or geometries of objects).

Each demonstration $m \in \{1, \dots, M\}$ contains T_m datapoints forming a dataset of N datapoints $\{\boldsymbol{\xi}_t\}_{t=1}^{N}$ with $N = \sum_m^M T_m$.

The task parameters are represented by P coordinate systems, defined at time step t by $\{\boldsymbol{b}_{t,j}, \boldsymbol{A}_{t,j}\}_{j=1}^{P}$, representing respectively the origin of the observer and a transformation matrix.

The demonstrations $\boldsymbol{\xi} \in \mathbb{R}^{D \times N}$ are observed from these different viewpoints, forming P trajectory samples $\boldsymbol{X}^{(j)} \in \mathbb{R}^{D \times N}$. These samples can be collected from sensors located at the frames, or computed with

$$\boldsymbol{X}_t^{(j)} = \boldsymbol{A}_{t,j}^{-1}(\boldsymbol{\xi}_t - \boldsymbol{b}_{t,j}). \tag{4}$$

The parameters of a TP-GMM with K components are defined by $\left\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\right\}_{i=1}^{K}$ (π_i are the mixing coefficients, $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Sigma}_i^{(j)}$ are the center and covariance matrix of the i-th Gaussian component in frame j).

Learning of the parameters is achieved by log-likelihood maximization subject to the constraint that the data in the different frames arose from the same source, resulting in an *expectation-maximization* (EM) algorithm [23] to iteratively update the model parameters until convergence, see Appendix A for details. Other forms of learning for mixture models are possible, including spectral clustering [68, 84, 52], online learning [67, 82, 27, 99, 34] or self-refinement [19].

For a movement in Cartesian space with 10 demonstrations and 3 candidate frames, the overall learning process typically takes 1-3 *sec* on a standard laptop. The reproduction is much faster and can be computed online (usually below 1 *msec*).

The learned model is then used to reproduce movements in other situations (for new position and orientation of candidate frames). A new GMM with parameters $\{\pi_i, \hat{\boldsymbol{\xi}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\}_{i=1}^{K}$ can automatically be generated with

$$\mathcal{N}\left(\hat{\boldsymbol{\xi}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\right) \propto \prod_{j=1}^{P} \mathcal{N}\left(\hat{\boldsymbol{\xi}}_{t,i}^{(j)},\ \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)}\right), \text{ with}$$
$$\hat{\boldsymbol{\xi}}_{t,i}^{(j)} = \boldsymbol{A}_{t,j}\boldsymbol{\mu}_i^{(j)} + \boldsymbol{b}_{t,j}\ , \quad \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)} = \boldsymbol{A}_{t,j}\boldsymbol{\Sigma}_i^{(j)}\boldsymbol{A}_{t,j}^{\top}, \tag{5}$$

where the result of the Gaussian product is given by

$$\hat{\boldsymbol{\Sigma}}_{t,i} = \Big(\sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)^{-1}}\Big)^{-1}, \quad \hat{\boldsymbol{\xi}}_{t,i} = \hat{\boldsymbol{\Sigma}}_{t,i} \sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)^{-1}} \hat{\boldsymbol{\xi}}_{t,i}^{(j)}. \tag{6}$$

For computational efficiency, the above equations can be computed with precision matrices instead of covariances. Fig. 4 depicts the different steps of the above computation.

The proposed task-parameterized approach requires each frame to evaluate the local variability of the demonstrations. This section showed that a mixture model could be employed to extract this variability. However, other encoding strategies can be used as long as the local variations take the form of full covariances aligned with the different frames. In particular, data-driven encoding strategies can alternatively be employed. This will be shown later in Section 8 by using two examples with *task-parameterized Gaussian process* (TP-GP) and *task-parameterized locally weighted regression* (TP-LWR).

A *Matlab/GNU Octave* implementation of TP-GMM can be found in the `demo_TPGMM01.m` example. An example with a standard mixture is also provided in `demo_GMM01.m`. An example showing the construction of frames and the collection of data in different frames is provided in `demo_affineTransform01.m`.

3.1 Regularization of the TP-GMM parameters

In applications that are prone to overfitting, it is relevant to introduce regularization terms. Regularization has the effect of avoiding singularities and smoothing the solution space. An option is to define a minimal admissible eigenvalue $\lambda_{\min}$ and adjust each covariance matrix $\boldsymbol{\Sigma}_i^{(j)}$ so that

$$\boldsymbol{\Sigma}_i^{(j)} \leftarrow \boldsymbol{V}_i^{(j)} \tilde{\boldsymbol{D}}_i^{(j)} \boldsymbol{V}_i^{(j)^{\top}}, \tag{7}$$
$$\text{with} \quad \tilde{\boldsymbol{D}}_i^{(j)} = \begin{bmatrix} \tilde{\lambda}_{i,1}^{2(j)} & 0 & \cdots & 0 \\ 0 & \tilde{\lambda}_{i,2}^{2(j)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\lambda}_{i,d}^{2(j)} \end{bmatrix},$$
$$\text{and} \quad \tilde{\lambda}_{i,k}^{(j)} = \max(\tilde{\lambda}_{i,k}^{(j)}, \lambda_{\min}) \quad \forall k \in \{1, \dots, d\},$$

where $\boldsymbol{V}_i^{(j)}$ is a matrix containing the stacked eigenvectors of $\boldsymbol{\Sigma}_i^{(j)}$, with $\lambda_{i,k}^{(j)}$ the corresponding eigenvalues.

Another approach is to set *a priori* uncertainties on the covariance parameters in the form of a diagonal isotropic covariance $\boldsymbol{I}\rho$ (Tikhonov regularization), so that

$$\boldsymbol{\Sigma}_i^{(j)} \leftarrow \boldsymbol{\Sigma}_i^{(j)} + \boldsymbol{I}\rho, \tag{8}$$

with $\boldsymbol{I}$ an identity matrix and ρ a small scalar factor that can be either set empirically or estimated from the data.

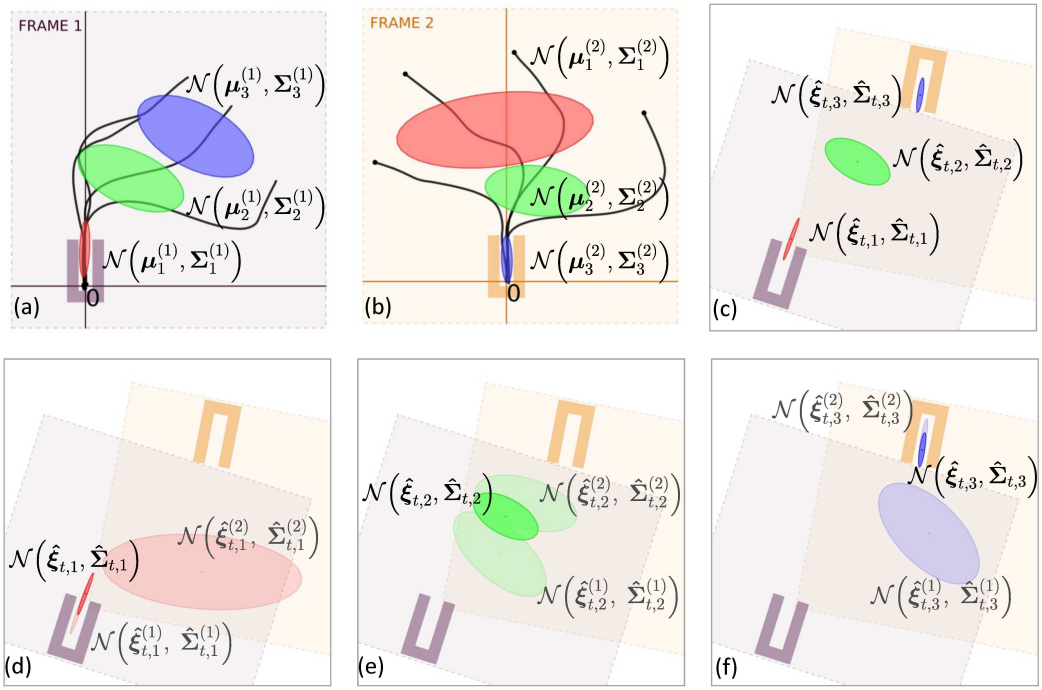


Figure 4: TP-GMM retrieval process and associated variables. *(a-b)* shows the model parameters (TP-GMM with 3 Gaussians and 2 frames). *(c)* Temporary GMM retrieved at time step t for a new configuration of the two frames. *(d-f)* Details of computation, where each temporary Gaussian is retrieved as a product of linearly transformed Gaussians.

The difference with Eq. (7) is that a value ρ is added to each $\lambda_{i,k}^{(j)}$ instead of truncating the eigenvalues. The same development can be done with singular value decomposition, emphasizing the effect of the regularization on the condition number, by forcing it to be higher than a threshold as in Eq. (7) or by increasing the singular values as in Eq. (8). It is in some applications convenient to apply small regularization terms at different steps of the procedure (e.g., at each iteration in the EM process and after convergence before computing Gaussian products).

4 Extension to task-parameterized subspace clustering

Classical Gaussian mixture models tend to perform poorly in high-dimensional spaces if too few datapoints are available. This is also true for robotics problems aiming at encoding multivariate and multimodal signals from only few demonstrations. Namely, if the training set is $\{\boldsymbol{\xi}_t\}_{t=1}^N$ with $\boldsymbol{\xi}_t \in \mathbb{R}^D$, the *curse of dimensionality* occurs if the dimension of the data D is too large compared to the size of the training set N. In particular, the problem can affect the full covariances $\boldsymbol{\Sigma}_i^{(j)} \in \mathbb{R}^{D\times D}$ in (52) because the number of parameters to be estimated quadratically grows with D.

Bouveyron and Brunet reviewed various ways of viewing the problem and coping with high-dimensional data in clustering problems [11]. In practice, three viewpoints can be considered:

1. Since D is too large compared to N, a global dimensionality reduction should be applied as a preprocessing step to reduce D.

2. Since D is too large compared to N, the solution space contains many poor local optima; the solution space should be smoothed by introducing ridge or lasso regularization in the estimation of the covariance (avoiding numerical problem and singular solutions when inverting the covariances). As discussed in Section 3.1, a simple form of regularization can be achieved after the maximization step of each EM loop.

3. Since D is too large compared to N, the model is probably over-parametrized, and a more parsimonious model should be used (thus estimating a fewer number of parameters).

One example falling in the last category would be to consider spherical or diagonal covariances instead of full matrices, corresponding to a separate treatment of each variable. Although commonly employed in robotics, such decoupling is a limiting factor to encode gestures and sensorimotor streams, because it does not fully exploit principles underlying coordination, motor skill acquisition and actionperception couplings [66, 41, 80, 53, 94, 86, 83, 103].

Our rationale is that diagonal constraints are too strong for motor skill encoding, because it loses important synergistic information among the variables. There are, however, a wide range of alternatives in mixture modeling, which are in-between the encoding of diagonal and full covariances, and that can readily be exploited in the context of robot skills acquisition. These alternatives can be studied as a subspace clustering problem, that aims at grouping the data such that they can be locally projected in a subspace of reduced dimensionality, thus helping the analysis of the local trend of the movement, while reducing the number of parameters to be estimated, and "locking"

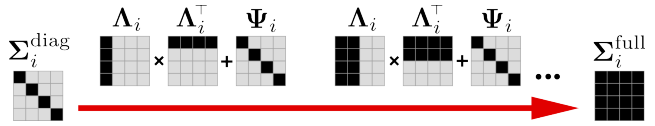


Figure 5: Exploitation of the covariance structure in a mixture of factor analyzers (MFA) to consider intermediary steps between the modeling as diagonal covariances (left) and full covariances (right).

the most important synergies to cope with perturbations.

Many possible constraints can be considered, grouped in families such as parsimonious GMM [11, 61, 7], *mixtures of factor analyzers* (MFA) [60] or *mixtures of probabilistic principal component analyzers* (MPPCA) [93]. These techniques will next be described in the context of task-parameterized models.

4.1 Parsimonious TP-GMM

By following the perspective of Section 3.1, a parsimonious TP-GMM can be defined by considering the spectral decomposition of the covariances

$$\boldsymbol{\Sigma}_i^{(j)} = \boldsymbol{V}_i^{(j)} \boldsymbol{D}_i^{(j)} \boldsymbol{V}_i^{(j)\top}, \tag{9}$$

with $\boldsymbol{V}_i^{(j)}$ a matrix of ordered eigenvectors (determining the orientation of the cluster) and $\boldsymbol{D}_i^{(j)}$ a diagonal matrix with ordered eigenvalues $\lambda_{i,k}^{(j)}$ (determining the shape of the cluster), where constraints are set by sharing some of these elements among the clusters, and/or by keeping only the first d eigenvectors and eigenvalues in the parameterization.

The *high-dimensional data clustering* (HDDC) approach from [12] lies in this category of models addressing both subspace clustering and regularization. An example of implementation is to consider that the subspace of each cluster i is generated by the first d_i eigenvectors associated with the first $d_i^{(j)}$ eigenvalues $\lambda_{i,k}^{(j)}$, and that outside of this subspace, the variance is spherical, modeled by a single parameter

$$\bar{\lambda}_i^{(j)} = \frac{1}{D-d_i^{(j)}} \sum_{k=d_i^{(j)}+1}^{D} \lambda_{i,k}^{(j)} = \frac{1}{D-d_i^{(j)}} \left(\text{tr}(\boldsymbol{\Sigma}_i^{(j)}) - \sum_{k=1}^{d_i^{(j)}} \lambda_{i,k}^{(j)} \right), \tag{10}$$

which is used to reconstruct a full covariance matrix by replacing the last $D - d_i^{(j)}$ eigenvalues with $\bar{\lambda}_i^{(j)}$.

A *Matlab/GNU Octave* implementation of HDDC in the context of task-parameterized models can be found in `demo_TPHDDC01.m`.
An example with a standard mixture models is also provided in `demo_HDDC01.m`.

4.2 Task-parameterized mixture of factor analyzers (TP-MFA)

Factor analysis (FA) is an approach as old as *principal component analysis* (PCA) to cope with dimension reduction, often overshadowed by PCA although is has an equivalently important literature on the topic [12]. The basic idea of factor analysis is to reduce the dimensionality of the data while keeping the observed covariance structure, see [26] for an example of application in robotics.

The TP-GMM presented in Section 3 is fully compatible with subspace clustering approaches based on factor analysis. A *task-parameterized mixture of factor analyzers* (TP-MFA) assumes for each component i and frame j a covariance structure of the form

$$\boldsymbol{\Sigma}_i^{(j)} = \boldsymbol{\Lambda}_i^{(j)} \boldsymbol{\Lambda}_i^{(j)\top} + \boldsymbol{\Psi}_i^{(j)}, \tag{11}$$

where $\boldsymbol{\Lambda}_i^{(j)} \in \mathbb{R}^{D \times d}$, known as the *factor loadings matrix*, typically has $d < D$ (providing a parsimonious representation of the data), and a diagonal noise matrix $\boldsymbol{\Psi}_i^{(j)}$.

The factor loading and noise terms of the covariance matrix can be constrained in different ways (e.g., such as being shared across Gaussian components), yielding a collection of eight parsimonious covariance structures [61]. For example, the *task-parameterized mixture of probabilistic principal component analyzers* (TP-MPPCA) [93] is a special case of TP-MFA with the distribution of the errors assumed to be isotropic with $\boldsymbol{\Psi}_i^{(j)} = \boldsymbol{I} {\sigma_i^{(j)}}^2$.

Fig. 5 shows that the covariance structure in MFA can span a wide range of covariances.

Appendix B details the structure of TP-MFA and provides an EM algorithm to estimate the model parameters.

The hypothesis of TP-MFA models can be viewed as less restrictive as TP-HDDC models based on eigendecomposition (see Section 4.1), because the subspace of each class does not need to be spanned by orthogonal vectors, whereas it is a necessary condition in models based on eigendecomposition [12].

Similarly to parsimonious GMM based on eigendecomposition, the covariances in TP-MFA can be constrained by fixing d or by sharing elements among the mixture components. This encoding strategy can then be extended to variants of MFA aiming at optimizing the sharing and reuse of subspaces among the Gaussian components, such as in semi-tied covariance [31]. These techniques can be exploited to extend the concept of synergies to a wider range of rich motor skills, with a simultaneous segmentation and re-use of previously discovered synergies.

For each approach, a dedicated EM update can be derived corresponding to the type of constraints considered [61]. They all reconstruct estimates of the full covariances, which is an important characteristic that will be exploited in the next sections of this article.

The TP-MFA extension of TP-GMM opens several roads for further investigation. Bayesian nonparametric approaches such as [101] can be used to simultaneously select the number of clusters and the dimension of the subspace in each cluster. Another extension is to use tied structures in the covariances to enable the organization and reuse of previously acquired synergies [31].

Another possible extension is to enable deep learning strategies in task-parameterized models. As discussed in [92], the prior of each FA can be replaced by a separate second-level MFA that learns to model the aggregated posterior of that FA (instead of the isotropic Gaussian), providing a hierarchical structure organization where one layer of latent variables can be learned at a time. This

Figure 6: Example of TP-MFA to encode and retrieve full-body dancing motion from [36]. Here, the motion of one of the two partners ($D=94$) is retrieved by adapting it online to the motion of the other partner. In the model, the red stick figure (in thin line) is the frame of reference and the gray stick figure (in thick line) is the motion encoded in TP-MFA. The black stick figure (in thick line) shows the motion regenerated with TP-MFA and Gaussian mixture regression, based on the observation of the red stick figure. For 12 Gaussian components ($K=12$) and a subspace of 2 dimensions ($d=2$) encoding a motion of 94 dimensions, the total number of parameters in TP-MFA is 4511. The corresponding number of parameters in a TP-GMM with full covariances would be 54719. We can see that TP-MFA generates a smooth and natural movement similar to the original dance.

can be exploited as a link with deep learning strategies [40, 9] for real-valued high-dimensional data within directed graphical models.

Fig. 6 shows a kinematic example with TP-MFA used for encoding and synthesis purposes.

Matlab/GNU Octave implementations of TP-MFA and TP-MPPCA can be found in `demo_TPMFA01.m` and `demo_TPMPPCA01.m`. The corresponding examples for standard mixture models can also be found in `demo_MFA01.m` and `demo_MPPCA01.m`.

5 Extension to motion synthesis

While the previous sections focused only on TP-GMM as an encoding strategy, this section addresses the problem of generating movements from the model.

Several approaches can be used to retrieve continuous movements from a TP-GMM. The next two subsections provide examples for two different synthesis techniques. The first technique is to encode a decay term or a time variable as an additional feature in the mixture, and use *Gaussian mixture regression* (GMR) [33] to retrieve movements adapted to the current situations. The second technique is to encode both static and dynamic features in the mixture model as in *trajectory-HMM* [30, 95, 106, 89]. These two techniques are described next.

5.1 Gaussian mixture regression (GMR)

With a GMM representation, the reproduction of a movement can be formalized as a regression problem [33]. We showed in [18, 17] that in robot learning, *Gaussian mixture regression* (GMR) offers a simple solution to generate continuous movements from a GMM. GMR relies on basic properties of normal distributions (linear transformation and conditioning). It provides a probabilistic retrieval of movements or policies, in which the model can compute the next actions on-the-fly, with a computation time that is independent of the number of datapoints used to train the model.

In contrast to other regression methods such as *locally weighted regression* (LWR) [78], *locally weighted projection regression* (LWPR) [100], or *Gaussian process regression* (GPR) [69, 35, 74], GMR does not model the regression function directly. It models the joint probability density function of the data, and then derives the regression function from the joint density model, see [88] for an excellent review of regression approaches. The estimation of the model parameters is thus achieved in an offline phase that depends linearly on the number of datapoints. Regression is then independent of this number and can be computed very rapidly, which makes the approach an interesting alternative to regression methods whose processing grows with the size of the dataset. The other benefit is that both input and output variables can be multidimensional without modification of the model.

GMR can for example be employed in robot applications requiring input and output dimensions to be specified at run time (e.g., to handle missing sensory inputs, or to react swiftly by retrieving partial outputs).

The superscripts $\mathcal{I}$ and $\mathcal{O}$ will be further used to describe the dimensions that span for input and output (used

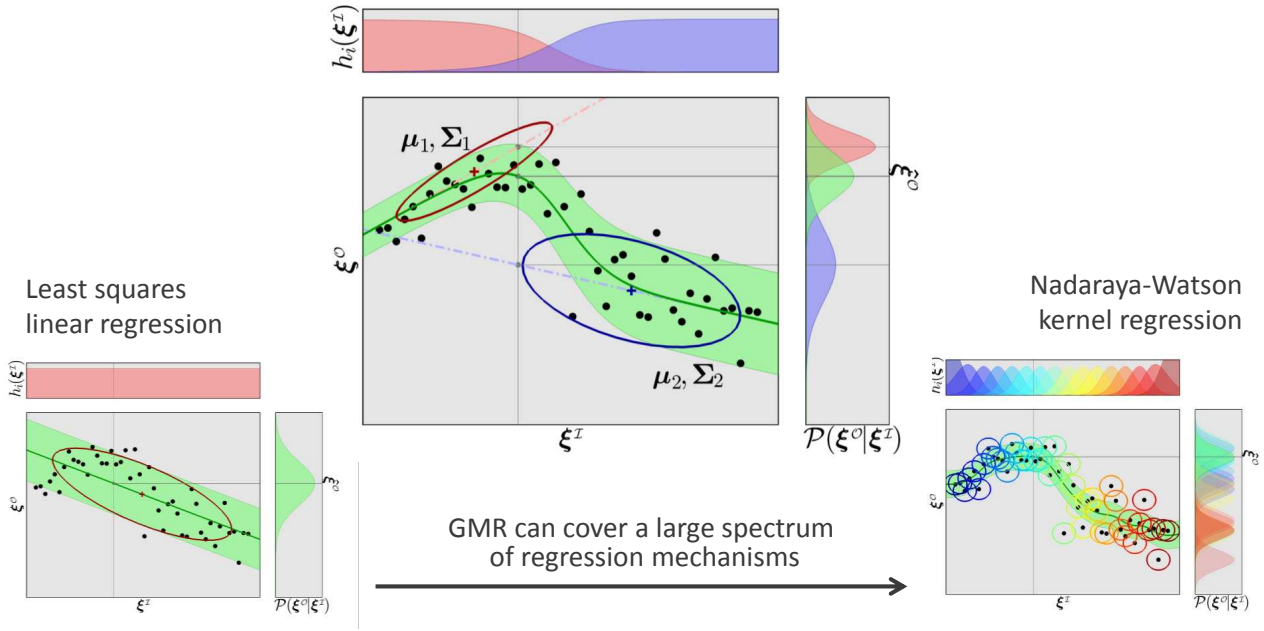


Figure 7: Illustration of the encoding of $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \boldsymbol{\xi}^{\mathcal{O}})$ as a Gaussian mixture model (GMM) with two components, and estimation of $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{O}}|\boldsymbol{\xi}^{\mathcal{I}})$ with Gaussian mixture regression (GMR), where both $\boldsymbol{\xi}^{\mathcal{I}}$ and $\boldsymbol{\xi}^{\mathcal{O}}$ can be multidimensional. The model can emulate a large spectrum of regression mechanisms, from standard linear regression (when a single component $K=1$ is used), to non-linear kernel regression (with $K=N$ and a Gaussian centered on each datapoint).

as exponents for vectors and matrices). The general case of a GMM encoding a dataset $\boldsymbol{\xi}$ with the joint distribution $\mathcal{P}(\boldsymbol{\xi}^{\mathcal{I}}, \boldsymbol{\xi}^{\mathcal{O}}) \sim \sum_{i=1}^{K} \pi_i \, \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ will first be described, which will later be extended to its task-parameterized version.

At each iteration step t, the datapoint $\boldsymbol{\xi}_t$ can be decomposed as two subvectors $\boldsymbol{\xi}_t^{\mathcal{I}}$ and $\boldsymbol{\xi}_t^{\mathcal{O}}$ spanning for the input and output dimensions. For trajectory encoding in task space, $\mathcal{I}$ corresponds to the time input dimension (e.g., value of a decay term), and $\mathcal{O}$ corresponds to the output dimensions describing a path (e.g., end-effector position in task space).

With this notation, a block decomposition of the datapoint $\boldsymbol{\xi}_t$, vectors $\boldsymbol{\mu}_i$ and matrices $\boldsymbol{\Sigma}_i$ can be written as

$$\boldsymbol{\xi}_t = \begin{bmatrix} \boldsymbol{\xi}_t^{\mathcal{I}} \\ \boldsymbol{\xi}_t^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathcal{I}} \\ \boldsymbol{\mu}_i^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{O}} \end{bmatrix}. \tag{12}$$

At each time step t during reproduction, $\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}})$ is computed as the conditional distribution

$$\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) \sim \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) \, \mathcal{N}\Big(\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}), \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big), \tag{13}$$

$$\text{with} \quad \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}) = \boldsymbol{\mu}_i^{\mathcal{O}} + \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} {\boldsymbol{\Sigma}_i^{\mathcal{I}}}^{-1} (\boldsymbol{\xi}_t^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}}), \tag{14}$$

$$\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} = \boldsymbol{\Sigma}_i^{\mathcal{O}} - \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} {\boldsymbol{\Sigma}_i^{\mathcal{I}}}^{-1} \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}}, \tag{15}$$

$$\text{and} \quad h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) = \frac{\pi_i \mathcal{N}(\boldsymbol{\xi}_t^{\mathcal{I}} | \, \boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}})}{\sum_k^K \pi_k \mathcal{N}(\boldsymbol{\xi}_t^{\mathcal{I}} | \, \boldsymbol{\mu}_k^{\mathcal{I}}, \boldsymbol{\Sigma}_k^{\mathcal{I}})}. \tag{16}$$

Note that Eq. (13) represents a multimodal distribution. For problems in which a single peaked output distribution is preferred, Eq. (13) can be approximated by a normal distribution (see Appendix C for details of computation)

$$\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \mathcal{N}\Big(\boldsymbol{\xi}_t^{\mathcal{O}} | \, \hat{\boldsymbol{\mu}}_t^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}}\Big), \quad \text{with} \tag{17}$$

$$\hat{\boldsymbol{\mu}}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) \, \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}), \tag{18}$$

$$\hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) \Big(\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} + \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}) \, \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}})^{\top}\Big) - \hat{\boldsymbol{\mu}}_t^{\mathcal{O}} \hat{\boldsymbol{\mu}}_t^{\mathcal{O}\top}. \tag{19}$$

The retrieved signal in Eq. (17) encapsulates variation and correlation information in the form of full covariance matrices. GMR has so far mostly been used in three manners:

1. as an autonomous system with $\boldsymbol{\xi} = [\boldsymbol{x}^{\top}, \dot{\boldsymbol{x}}^{\top}]^{\top}$, by learning $\mathcal{P}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ with a GMM, with $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$ representing position and velocity of the system (either in task space or joint space), and by retrieving iteratively during reproduction a series of velocity commands by estimating $\mathcal{P}(\dot{\boldsymbol{x}}|\boldsymbol{x})$ with GMR [38, 39, 17, 46];

2. as time-indexed trajectories with $\boldsymbol{\xi} = [t, \boldsymbol{x}^{\top}]^{\top}$, by learning $\mathcal{P}(t, \boldsymbol{x})$ with a GMM, and retrieving $\mathcal{P}(\boldsymbol{x}|t)$ with GMR for each time step to reproduce smooth trajectories (infinitely differentiable) [18].

3. as a probabilistic formulation of dynamic movement primitives (DMP) [20].

Alternatively, any subset of input-output dimensions can be selected, which can change, if required, at each iteration during reproduction. It can for example handle different sources of missing data, as the system is able to consider any combination of multidimensional input/output

mappings during the retrieval phase. Expectations on the remaining dimensions can be computed within the control loop of the robot, corresponding to a convex sum of linear approximations (with weights varying non-linearly).

Fig. 7 depicts the use of GMR in the case of time-indexed trajectories.

GMR can be viewed as a trade-off between a global and local approach in the sense that the placement and spread of the basis functions are learned, together with their responses, as a soft partitioning problem through *expectation-maximization* (EM),[1] while the prediction is a weighted superposition of locally linear systems. It provides variation and correlation information for the retrieved multidimensional output, enabling the extraction of local coordination patterns in the movement.

Note that if the application requires the encoding of high-dimension data from few observations, subspace learning techniques such as MFA (see Section 4) can be used jointly with GMR to locally reduce the dimensionality without modifying the regression process.

The combination of TP-GMM and GMR is simply achieved by augmenting the dataset in each frame with an input dimension, and defining all task parameters $\boldsymbol{A}_{t,j}$ and $\boldsymbol{b}_{t,j}$ so that the input is not modulated by the task parameterization. Compared to an initial TP-GMM encoding $\boldsymbol{\xi}^{\mathcal{O}}$ with task parameters $\boldsymbol{A}^{\mathcal{O}}_{t,j}$ and $\boldsymbol{b}^{\mathcal{O}}_{t,j}$, the combination of TP-GMM and GMR instead encodes $\boldsymbol{\xi}=\left[\boldsymbol{\xi}^{\mathcal{I}\top},\boldsymbol{\xi}^{\mathcal{O}\top}\right]^\top$ with task parameters

$$\boldsymbol{A}_{t,j}=\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{A}^{\mathcal{O}}_{t,j}\end{bmatrix},\quad \boldsymbol{b}_{t,j}=\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{b}^{\mathcal{O}}_{t,j}\end{bmatrix}, \tag{20}$$

where in the case of a decay term (or an explicit time variable driving the system), the identity matrix $\boldsymbol{I}$ collapses to 1.[2]

A *Matlab/GNU Octave* implementation of TP-GMM with GMR can be found in `demo_TPGMR01.m`. The corresponding example for standard mixture models can also be found in `demo_GMR01.m`. Examples of DMP learning with GMR can be found in `demo_DMP_GMR*.m`.

5.2 GMM with dynamic features (trajectory-GMM)

In the field of speech processing, the extraction of statistics from both static and dynamic features within a *hidden Markov model* (HMM) has a long history [30, 95, 106]. In particular, it can be used in speech synthesis to avoid discontinuities in the generated speech spectra. The synthesized speech then becomes natural and smooth even when a small number of Gaussians is used. This is achieved by coordinating the distributions of both static and dynamic features (the dynamic features are often called delta and delta-delta parameters). In speech processing, these parameters usually corresponds to the evolution of mel-frequency cepstral coefficients characterizing the power spectrum of a sound, but the same *trajectory-HMM* approach can be used with any form of continuous signals. In robotics, this approach has rarely been exploited, at the exception of the work from Sugiura *et al.* employing it to represent object manipulation movements [89].

For the encoding of movements, velocity and acceleration can alternatively be used as dynamic features. By considering an Euler approximation, the velocity is computed as

$$\dot{\boldsymbol{x}}_t=\frac{\boldsymbol{x}_{t+1}-\boldsymbol{x}_t}{\Delta t}, \tag{21}$$

where $\boldsymbol{x}_t$ is a multivariate position vector. The acceleration is similarly computed as

$$\ddot{\boldsymbol{x}}_t=\frac{\dot{\boldsymbol{x}}_{t+1}-\dot{\boldsymbol{x}}_t}{\Delta t}=\frac{\boldsymbol{x}_{t+2}-2\boldsymbol{x}_{t+1}+\boldsymbol{x}_t}{\Delta t^2}. \tag{22}$$

By using (21) and (22), a vector $\boldsymbol{\zeta}_t$ will be used to represent the concatenated position, velocity and acceleration vectors at time step t, namely[3]

$$\boldsymbol{\zeta}_t=\begin{bmatrix}\boldsymbol{x}_t\\ \dot{\boldsymbol{x}}_t\\ \ddot{\boldsymbol{x}}_t\end{bmatrix}=\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}\\ -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0}\\ \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}_t\\ \boldsymbol{x}_{t+1}\\ \boldsymbol{x}_{t+2}\end{bmatrix}. \tag{23}$$

$\boldsymbol{\zeta}$ and $\boldsymbol{x}$ are then defined as large vectors concatenating $\boldsymbol{\zeta}_t$ and $\boldsymbol{x}_t$ for all time steps, namely

$$\boldsymbol{\zeta}=\begin{bmatrix}\boldsymbol{\zeta}_1\\ \boldsymbol{\zeta}_2\\ \vdots\\ \boldsymbol{\zeta}_T\end{bmatrix},\quad \boldsymbol{x}=\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\\ \vdots\\ \boldsymbol{x}_T\end{bmatrix}. \tag{24}$$

Similarly to the matrix operator (23) defined for a single time step, a large sparse matrix $\boldsymbol{\Phi}$ can be defined so that $\boldsymbol{\zeta}=\boldsymbol{\Phi}\boldsymbol{x}$, namely[4]

$$\overbrace{\begin{bmatrix}\vdots\\ \boldsymbol{x}_t\\ \dot{\boldsymbol{x}}_t\\ \ddot{\boldsymbol{x}}_t\\ \boldsymbol{x}_{t+1}\\ \dot{\boldsymbol{x}}_{t+1}\\ \ddot{\boldsymbol{x}}_{t+1}\\ \vdots\end{bmatrix}}^{\boldsymbol{\zeta}}=\overbrace{\begin{bmatrix}\ddots & \vdots & \vdots & \vdots & \iddots\\ \cdots & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \cdots\\ \cdots & -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0} & \cdots\\ \cdots & \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I} & \cdots\\ & \cdots & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}\\ & \cdots & -\frac{1}{\Delta t}\boldsymbol{I} & \frac{1}{\Delta t}\boldsymbol{I} & \boldsymbol{0}\\ & \cdots & \frac{1}{\Delta t^2}\boldsymbol{I} & -\frac{2}{\Delta t^2}\boldsymbol{I} & \frac{1}{\Delta t^2}\boldsymbol{I}\end{bmatrix}}^{\boldsymbol{\Phi}}\overbrace{\begin{bmatrix}\vdots\\ \boldsymbol{x}_t\\ \boldsymbol{x}_{t+1}\\ \boldsymbol{x}_{t+2}\\ \boldsymbol{x}_{t+3}\\ \vdots\end{bmatrix}}^{\boldsymbol{x}}. \tag{25}$$

The dataset $\{\boldsymbol{\zeta}_t\}_{t=1}^N$ with $N=\sum_m^M T_m$ is composed of M trajectory samples, where the m-th trajectory sample has T_m datapoints. It can be encoded in a *Gaussian mixture model* (GMM), *hidden Markov model* (HMM) or *hidden semi-Markov model* (HSMM) [72]. The example of a GMM encoding $\mathcal{P}(\boldsymbol{\zeta})\sim\sum_{i=1}^K\pi_i\,\mathcal{N}(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$ will be described here, which will later be extended to its task-parameterized version.

[1] Competition/collaboration arises due to the weighting term $h_{t,i}$ in Eq. (49) summing over the influence of the other Gaussian components.

[2] Possible extensions are possible here for a local modulation of movement duration.

[3] To simplify the notation, the number of derivatives will be set up to acceleration ($C=3$), but the results can easy be generalized to a higher or lower number of derivatives (in the provided source codes, a parameter automatically sets the number of derivatives to be considered).

[4] Note that a similar operator is defined to handle border conditions, and that $\boldsymbol{\Phi}$ can automatically be constructed through the use of Kronecker products, see source codes for details.

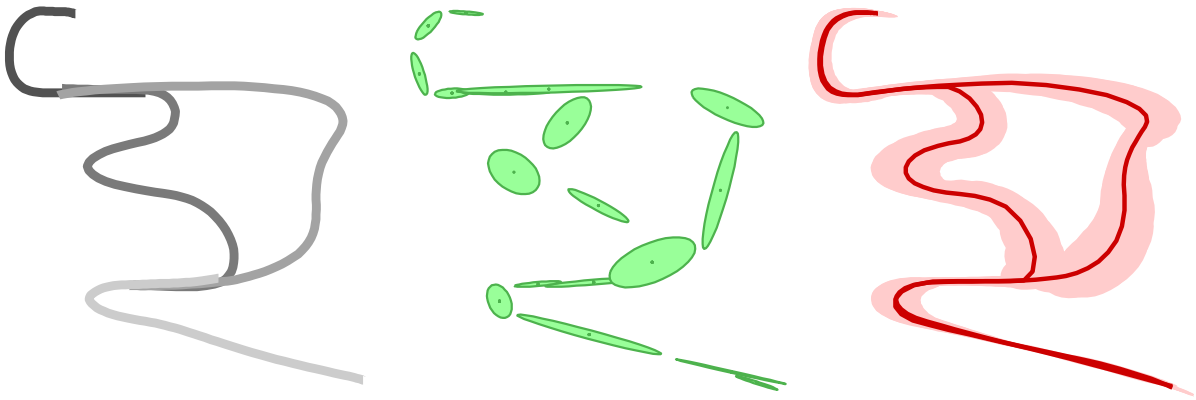


Figure 8: Example of trajectory-GMM encoding and retrieval. The planar motion contains multiple options, and is learned from a set of partial demonstrations that can be provided in any order. *Left:* Four demonstrations (represented with different shades of gray), corresponding to different subparts of a longer movement, where a part in the movement contains two optional paths. *Center:* The four demonstrations are used to train a trajectory-GMM with $K = 18$ components. *Right:* Two movements retrieved from the trajectory-GMM by stochastic sampling (with equal chance to take one or the other path). We can see that the movements are smooth, with an average position and full covariance estimated at each time step (represented as a light red flow tube of one standard deviation).

After training, for a given sequence of states $\boldsymbol{s} = \{s_1, s_2, \dots, s_T\}$ of T time steps, with discrete states $s_t \in \{1, \dots, K\}$,[5] the likelihood of a movement $\boldsymbol{\zeta}$ is given by

$$\mathcal{P}(\boldsymbol{\zeta}|\boldsymbol{s}) = \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\zeta}_t | \boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t}), \tag{26}$$

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 the conditional distribution

$$\mathcal{P}(\boldsymbol{\zeta}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\zeta}|\boldsymbol{\mu}_{\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{s}}), \tag{27}$$

with $\boldsymbol{\mu}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\mu}_{s_1} \\ \boldsymbol{\mu}_{s_2} \\ \vdots \\ \boldsymbol{\mu}_{s_T} \end{bmatrix}$ and $\boldsymbol{\Sigma}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\Sigma}_{s_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Sigma}_{s_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\Sigma}_{s_T} \end{bmatrix}$.

By using the relation $\boldsymbol{\zeta} = \boldsymbol{\Phi}\boldsymbol{x}$, we then seek during reproduction for a trajectory $\boldsymbol{x}$ maximizing (27), namely

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} \ \log \ \mathcal{P}(\boldsymbol{\Phi}\boldsymbol{x} \mid \boldsymbol{s}). \tag{28}$$

The part of $\log \ \mathcal{P}(\boldsymbol{\Phi}\boldsymbol{x} \mid \boldsymbol{s})$ dependent on $\boldsymbol{x}$ takes the quadratic error form

$$\begin{aligned} c &= (\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\zeta})^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} (\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\zeta}) \\ &= (\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\Phi}\boldsymbol{x})^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} (\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\Phi}\boldsymbol{x}). \end{aligned} \tag{29}$$

A solution can be found by differentiating the above objective function with respect to $\boldsymbol{x}$ and equating to 0, providing the trajectory (in vector form)

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \boldsymbol{\mu}_{\boldsymbol{s}}, \tag{30}$$

with the covariance error of the weighted least squares estimate given by

$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}} = \sigma \left(\boldsymbol{\Phi}^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \boldsymbol{\Phi}\right)^{-1}, \tag{31}$$

where σ is a scale factor.[6]

The resulting Gaussian $\mathcal{N}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}})$ forms a trajectory distribution. Other forms of trajectory distributions can be employed, where the main differences lie in the structure given to $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}}$ and in the way the basis functions are defined. A relevant example is the probabilistic movement primitives approach proposed by Paraschos *et al.* [71]. The structure of the trajectory distribution defined in [71] requires multiple trajectory demonstrations to avoid overfitting, but the problem can be circumvented by employing factorization and variational inference techniques [76].

In trajectory-GMM, the problem of setting the shape and spread of the basis functions, as well as the problem of determining the sparse structure of $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}}$ are directly framed within the GMM likelihood maximization problem, allowing the use of an EM algorithm to automatically organize the basis functions and find an appropriate structure for $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}}$, which will for example result in a sparse band-diagonal structure when the components are sufficiently decoupled, and which will account for the correlations within $\boldsymbol{\zeta}_t$.

An illustration of the trajectory-GMM properties that are of interest in robotics are shown in Fig. 8.

The combination of TP-GMM and trajectory-GMM is achieved by augmenting the position data with its derivatives (e.g., with velocity and acceleration), and defining all task parameters $\boldsymbol{A}_{t,j}$ and $\boldsymbol{b}_{t,j}$ so that they also apply to the derivatives. Compared to an initial TP-GMM encoding $\boldsymbol{x}$ with task parameters $\boldsymbol{A}_{t,j}^{\mathcal{O}}$ and $\boldsymbol{b}_{t,j}^{\mathcal{O}}$, the combination of TP-GMM and trajectory-GMM instead encodes

[5] The use of an HSMM encoding can autonomously regenerate such sequence in a stochastic manner, which is not described here due to space constraints.

[6] Equations (30) and (31) describe a trajectory distribution, and can be computed efficiently with Cholesky and/or QR decompositions by exploiting the positive definite symmetric band structure of the matrices, see for example [87]. With the Cholesky decomposition $(\boldsymbol{\Sigma}^{\boldsymbol{s}})^{-1} = \boldsymbol{T}^\top \boldsymbol{T}$, the objective function is maximized when $\boldsymbol{T}\boldsymbol{\Phi}\boldsymbol{x} = \boldsymbol{T}\boldsymbol{\mu}^{\boldsymbol{s}}$. With a QR decomposition $\boldsymbol{T}\boldsymbol{\Phi} = \boldsymbol{Q}\boldsymbol{R}$, the equation becomes $\boldsymbol{Q}\boldsymbol{R}\boldsymbol{x} = \boldsymbol{T}\boldsymbol{\mu}^{\boldsymbol{s}}$ with a solution efficiently computed with $\boldsymbol{x} = \boldsymbol{R}^{-1}\boldsymbol{Q}^\top \boldsymbol{T}\boldsymbol{\mu}^{\boldsymbol{s}}$. When using *Matlab*, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}}$ in Equations (30) and (31) can for example be computed with the LSCOV function.

$\boldsymbol{\zeta} = \left[\boldsymbol{x}^\top, \dot{\boldsymbol{x}}^\top, \ddot{\boldsymbol{x}}^\top\right]^\top$ with task parameters

$$\boldsymbol{A}_{t,j} = \begin{bmatrix} \boldsymbol{A}^{\mathcal{O}}_{t,j} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^{\mathcal{O}}_{t,j} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}^{\mathcal{O}}_{t,j} \end{bmatrix}, \quad \boldsymbol{b}_{t,j} = \begin{bmatrix} \boldsymbol{b}^{\mathcal{O}}_{t,j} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}. \tag{32}$$

A *Matlab/GNU Octave* implementation of TP-GMM with trajectory-GMM can be found in the example `demo_TPtrajGMM01.m`.
The corresponding example for standard mixture models can also be found in `demo_trajGMM01.m` and `demo_trajMFA01.m`.

5.3 Dynamic-based *vs.* time-based features in GMM

GMR with time-indexed trajectories (Section 5.1) offers a simple solution for the generation of trajectories from a GMM, with the disadvantage that time-based GMR often requires in practice a preprocessing step such as *dynamic time warping* (DTW) to re-align multiple demonstrations in time.

For the same reason, time-based GMR may also reduce the applicability of motion synthesis to more complex forms of teaching interactions, such as learning recurring patterns (combination of discrete and periodic motions), or demonstrating different options in a movement. This is not the case for trajectory-GMM that can handle these two issues without further modification of the model (see Section 5.2)). This is achieved at the expense of increasing the GMM dimensionality (by encoding position, velocity and acceleration instead of position and time as in time-based GMR).

Another advantage of encoding dynamic features over time features is that partial demonstrations can easily be used in the training set, see Fig. 8. In service robotics applications, it can sometimes be difficult and inefficient to demonstrate a complex manipulation task in a single shot. A more user-friendly way would be to provide incremental corrections or piecewise demonstrations of whole body movements, where the models can be trained with partial chunks of the whole movement, see e.g. [54]. This is also required in kinesthetic teaching with robots endowed with a high number of articulations (since the user cannot control all degrees of freedom at the same time with two hands). Trajectory-GMM provides here a way to handle partial demonstrations without further modification of the model.

6 Extension to minimal intervention control

Previous sections discussed the problem of generating a reference trajectory that can adapt to the current situation, by assuming that a controller is available to track the retrieved reference trajectory. In this section, the problem is extended to that of directly finding a controller to reproduce the movement.

Section 2.2 showed that the objective function (3) underlying TP-GMM aims at finding points $\boldsymbol{\xi}_t$ minimizing a weighted sum of quadratic error terms, whose result corresponds to a product of Gaussians. A similar function can be defined for the search of a controller, whose objective is to find a feedforward and feedback policy (instead of finding a reference trajectory).

This section depicts the canonical problem of searching a controller $\boldsymbol{u}_t$ for the discrete linear dynamical system (double integrator)

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_{t+1} \\ \dot{\boldsymbol{x}}_{t+1} \end{bmatrix}}_{\boldsymbol{\xi}_{t+1}} = \underbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{I}\Delta t \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \boldsymbol{x}_t \\ \dot{\boldsymbol{x}}_t \end{bmatrix}}_{\boldsymbol{\xi}_t} + \underbrace{\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}\Delta t \end{bmatrix}}_{\boldsymbol{B}} \boldsymbol{u}_t. \tag{33}$$

minimizing the cost

$$\begin{aligned} C &= \left(\hat{\boldsymbol{\xi}}_T - \boldsymbol{\xi}_T\right)^\top \boldsymbol{\Sigma}_{s_T}^{-1} \left(\hat{\boldsymbol{\xi}}_T - \boldsymbol{\xi}_T\right) \\ &\quad + \sum_{t=1}^{T-1} \left(\left(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t\right)^\top \boldsymbol{\Sigma}_{s_t}^{-1} \left(\hat{\boldsymbol{\xi}}_t - \boldsymbol{\xi}_t\right) \; + \; \boldsymbol{u}_t^\top \boldsymbol{R}_t \, \boldsymbol{u}_t \right) \\ &= \left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\zeta}\right)^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} \left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\zeta}\right) \; + \; \boldsymbol{U}^\top \tilde{\boldsymbol{R}} \boldsymbol{U}, \end{aligned} \tag{34}$$

with $\boldsymbol{\mu}_{\boldsymbol{s}} \in \mathbb{R}^{TCD}$ and $\boldsymbol{\Sigma}_{\boldsymbol{s}} = \text{blockdiag}(\boldsymbol{\Sigma}_{s_1}, \boldsymbol{\Sigma}_{s_2}, \dots, \boldsymbol{\Sigma}_{s_T})$ with $\boldsymbol{\Sigma}_{\boldsymbol{s}} \in \mathbb{R}^{TCD \times TCD}$ defined as in Eq. (27), and $\tilde{\boldsymbol{R}} = \text{blockdiag}(\boldsymbol{R}, \boldsymbol{R}, \dots, \boldsymbol{R})$ with $\tilde{\boldsymbol{R}} \in \mathbb{R}^{(T-1)D \times (T-1)D}$ an additional 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 (29) used in the context of GMM with dynamic features (see Section 5.2) is a special case of (34) with $\tilde{\boldsymbol{R}} = \boldsymbol{0}$.

We showed in [16] that TP-GMM can be used to find a controller autonomously regulating the stiffness and damping behavior of the robot, see also Fig. 2-(d). The model shares links with optimal feedback control strategies in which deviations from an average trajectory are corrected only when they interfere with task performance, resulting in a controller satisfying *minimal intervention principle* [94, 103]. The approach also shares similarities with the solution proposed by Medina *et al.* in the context of risk-sensitive control for haptic assistance [62], by exploiting the predicted variability to form a minimal intervention controller (in task space or in joint space). The retrieved variability and correlation information is exploited to generate safe and natural movements within an optimal control strategy, in accordance to the predicted range of motion that could correctly reproduce the task in the current situation. Indeed, we demonstrated in [16] that TP-GMM is fully compatible with *linear quadratic regulation* (LQR) strategies, providing a controller adapted to the current situation with both impedance gains and reference trajectories varying with respect to external task parameters.

The tracking problem can be solved by different techniques, either exploiting tools from physics, dynamic programming or linear algebra [6, 10]. It can for example be solved with a batch approach by expressing all future states $\boldsymbol{x}_t$ as explicit function of the state $\boldsymbol{x}_1$. By writing

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

in a matrix form, we get

$$\underbrace{\begin{bmatrix}\boldsymbol{\xi}_1\\ \boldsymbol{\xi}_2\\ \boldsymbol{\xi}_3\\ \vdots\\ \boldsymbol{\xi}_T\end{bmatrix}}_{\boldsymbol{\zeta}} = \underbrace{\begin{bmatrix}\boldsymbol{I}\\ \boldsymbol{A}\\ \boldsymbol{A}^2\\ \vdots\\ \boldsymbol{A}^T\end{bmatrix}}_{\boldsymbol{S}^{\boldsymbol{\xi}}} \boldsymbol{\xi}_1 + \underbrace{\begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0}\\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & \boldsymbol{0}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{A}^{T-1}\boldsymbol{B} & \boldsymbol{A}^{T-2}\boldsymbol{B} & \cdots & \boldsymbol{B}\end{bmatrix}}_{\boldsymbol{S}^{\boldsymbol{u}}} \underbrace{\begin{bmatrix}\boldsymbol{u}_1\\ \boldsymbol{u}_2\\ \vdots\\ \boldsymbol{u}_{T-1}\end{bmatrix}}_{\boldsymbol{U}}, \tag{35}$$

with $\boldsymbol{\zeta} \in \mathbb{R}^{TCD}$, $\boldsymbol{S}^{\boldsymbol{\xi}} \in \mathbb{R}^{TCD\times CD}$, $\boldsymbol{\xi}_1 \in \mathbb{R}^{CD}$, $\boldsymbol{S}^{\boldsymbol{u}} \in \mathbb{R}^{TCD\times(T-1)D}$ and $\boldsymbol{U}\in\mathbb{R}^{(T-1)D}$.

Substituting (35) into (34), we get the cost function

$$C = \left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{S}^{\boldsymbol{\xi}}\boldsymbol{\xi}_1 - \boldsymbol{S}^{\boldsymbol{u}}\boldsymbol{U}\right)^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{S}^{\boldsymbol{\xi}}\boldsymbol{\xi}_1 - \boldsymbol{S}^{\boldsymbol{u}}\boldsymbol{U}\right) + \boldsymbol{U}^\top\tilde{\boldsymbol{R}}\boldsymbol{U}. \tag{36}$$

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

$$\hat{\boldsymbol{U}} = \left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\boldsymbol{S}^{\boldsymbol{u}} + \tilde{\boldsymbol{R}}\right)^{-1}\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{S}^{\boldsymbol{\xi}}\boldsymbol{\xi}_1\right), \tag{37}$$

corresponding to a damped weighted least squares estimate (ridge regression).

The sequence of acceleration commands (37) can either be used as a planning technique to reconstruct a trajectory with (35), or as a control technique to estimate feedforward and feedback terms for the current iteration.

The same controller can also be found iteratively with a Riccati equation, see [16] for details.

It is worth noting that the constraint (33) defines the same set of relations as (21) and (22) used in trajectory-GMM (GMM with dynamic features). The main difference between the two problems is that trajectory-GMM seeks for a reference trajectory, while the above problem seeks for a controller. Both problems results in a weighted least squares solution, with the difference that (37) uses a Tikhonov regularization term corresponding to the cost $\boldsymbol{R}$ that we set on the control inputs.

Matlab/GNU Octave implementations of GMM combined with LQR can be found in the examples `demo_batchLQR01.m` and `demo_iterativeLQR01.m`.
Additional examples can be found in `demo_testLQR01.m` - `demo_testLQR04.m`.

Extension to multiple coordinate systems

The above optimal control problem can be extended to reference trajectories expressed in P coordinate systems. By extending the cost in (34) to

$$\tilde{C} = \sum_{j=1}^{P}\left(\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)} - \boldsymbol{\zeta}\right)^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{(j)^{-1}}\left(\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)} - \boldsymbol{\zeta}\right) + \boldsymbol{U}^\top\tilde{\boldsymbol{R}}\boldsymbol{U}, \tag{38}$$

the sequence of control inputs becomes

$$\hat{\boldsymbol{U}} = \left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\boldsymbol{S}^{\boldsymbol{u}} + \tilde{\boldsymbol{R}}\right)^{-1}\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{S}^{\boldsymbol{\xi}}\boldsymbol{\xi}_1\right),$$
$$\text{with}\quad \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1} = \sum_{j=1}^{P}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{(j)^{-1}},\ \boldsymbol{\mu}_{\boldsymbol{s}} = \boldsymbol{\Sigma}_{\boldsymbol{s}}\sum_{j=1}^{P}\boldsymbol{\Sigma}_{\boldsymbol{s}}^{(j)^{-1}}\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)}.$$

whose intermediary variables $\boldsymbol{\mu}_{\boldsymbol{s}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{s}}$ correspond to the Gaussian resulting from the product of Gaussians with centers $\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)}$ and covariances $\boldsymbol{\Sigma}_{\boldsymbol{s}}^{(j)}$. Namely, solving

$$\hat{\boldsymbol{U}} = \arg\min_{\boldsymbol{U}}\sum_{j=1}^{P}\left(\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)} - \boldsymbol{\zeta}\right)^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{(j)^{-1}}\left(\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)} - \boldsymbol{\zeta}\right) + \boldsymbol{U}^\top\tilde{\boldsymbol{R}}\boldsymbol{U}$$

is equivalent to the two step optimization process

$$\boldsymbol{\mu}_{\boldsymbol{s}} = \arg\min_{\boldsymbol{\zeta}}\sum_{j=1}^{P}\left(\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)} - \boldsymbol{\zeta}\right)^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{(j)^{-1}}\left(\boldsymbol{\mu}_{\boldsymbol{s}}^{(j)} - \boldsymbol{\zeta}\right),$$
$$\hat{\boldsymbol{U}} = \arg\min_{\boldsymbol{U}}\left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\zeta}\right)^\top \boldsymbol{\Sigma}_{\boldsymbol{s}}^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{s}} - \boldsymbol{\zeta}\right) + \boldsymbol{U}^\top\tilde{\boldsymbol{R}}\boldsymbol{U},$$

showing that the combination of TP-GMM with LQR corresponds to an optimal controller acting in multiple coordinate systems.

The problem can be viewed as a form of *inverse optimal control* (IOC) [1, 2, 24], or more precisely, as a rudimentary form of IOC which can be solved analytically. Namely, it can provide a controller without exploratory search, at the expense of being restricted to simple forms of objectives (weighted sums of quadratic errors whose weights are learned from the demonstrations). This dual view can be exploited in further research to bridge action-level and goal-level imitation, or to provide better initial estimates in IOC problems.

Fig. 9 shows that a TP-GMM with a single Gaussian, combined with a minimal intervention controller based on LQR, can be used to encode and retrieve various behaviors.

Matlab/GNU Octave implementations of TP-GMM combined with linear quadratic optimal control can be found in `demo_TPbatchLQR01.m` and `demo_TPbatchLQR02.m`.
Additional examples with iterative computation (with finite or infinite horizon) instead of batch computation can be found in `demo_TPGMR_LQR01.m` and `demo_TPGMR_LQR02.m`, with as baseline comparison in `demo_TPGMR_DS01.m` (controller with predefined gains).

7 Extension to task parameters in the form of projection constraints

Thus far, this tutorial considered problems in which the task parameters were related to position, orientation or shape of objects in a Cartesian space. However, the use of TP-GMM is not limited can be extended to other forms of locally linear transformations or projections. The consideration of non square $\boldsymbol{A}_{t,j}$ matrices is for example relevant for the consideration of soft constraints in both configuration and operational spaces (through Jacobian operators), see [15] for a preliminary work in this direction.

It can also provide a principled way to learn nullspace constraints in a probabilistic form. The different frames correspond in this case to various candidate subspace projections of the movement, with statistical information extracted from the different projections.

An important and challenging category of applications include the problems requiring priority constraints [96, 57,

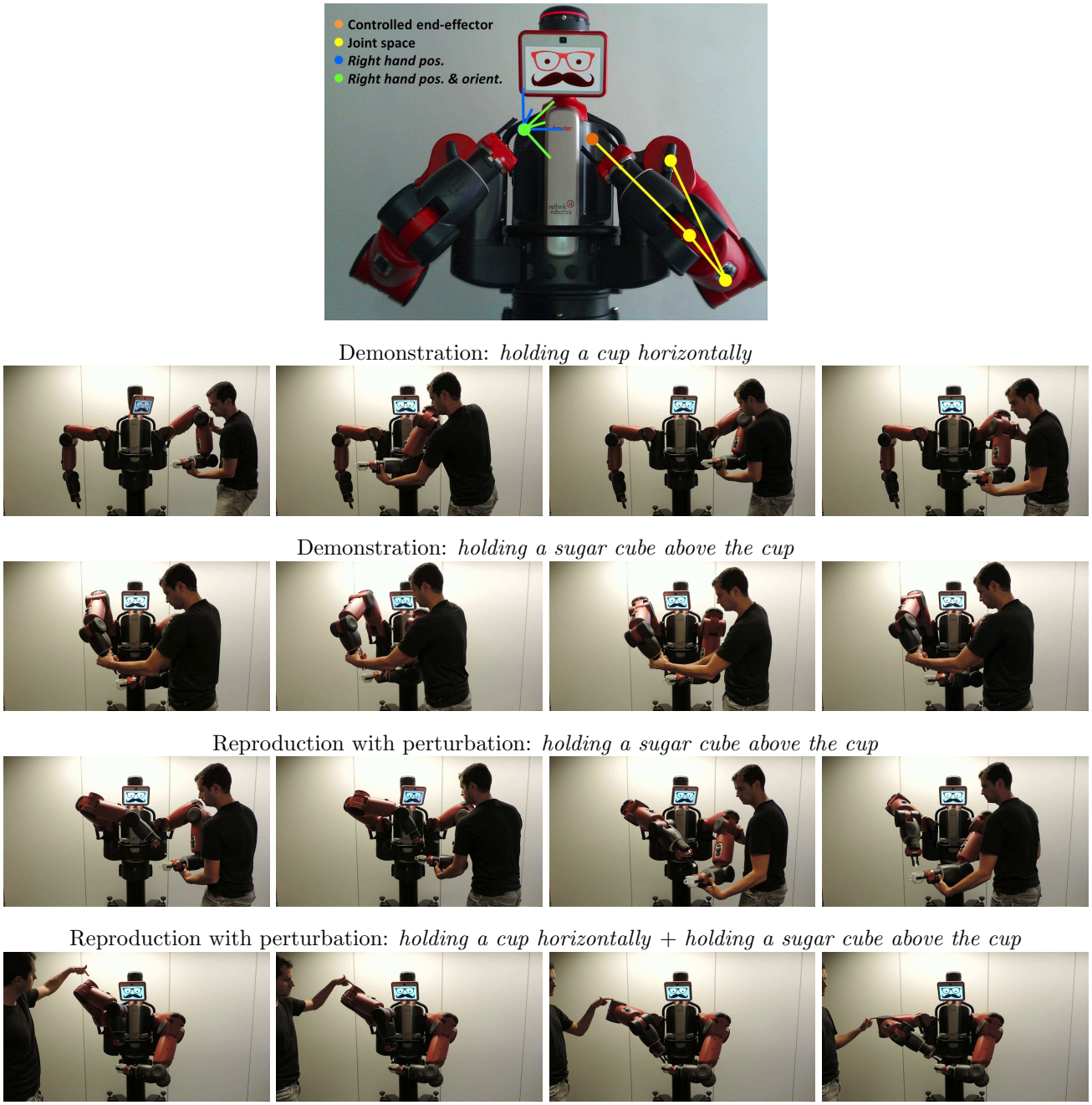


Figure 9: Learning of two behaviors with the Baxter robot at Idiap. The taught tasks consist of holding a cup horizontally with one hand, and holding a sugar cube above the cup with the other hand. The demonstrations are provided in two steps by kinesthetic teaching, namely, by holding the arms of the robot and moving them through the task while the robot compensates for the effect of gravity on its limbs. This procedure allows the user to move the robot arms without feeling their weight and without feeling the motors in the articulations, while the sensors are used to record the movement. Here, the data are recorded in several frames of reference (*top image*). During reproduction, the robot is controlled by following a minimal intervention principle, where the impedance parameters of the robot (stiffness and damping of a virtual spring pulling the robot arms) are automatically set in accordance to the extracted variation and coordination patterns. *First sequence:* Brief demonstration to show the robot how to hold a cup horizontally. *Second sequence:* Brief demonstration to show how to hold a sugar cube above the cup. *Third sequence:* Manual displacement of the left arm to test the learned behavior (the coordination of the two hands is successfully retrieved). *Last sequence:* Combination of the two learned tasks within a minimal intervention controller. Here, the user pushes the robot to show that the robot remains soft for perturbations that do not conflict with the acquired task constraints (automatic exploitation of the redundant degrees of freedom that do not conflict with the task).

37, 104, 77]. Such constraints can be learned and encoded within a TP-GMM from an initial set of task hierarchies given as potential candidates to describe the observed skill. The probabilistic encoding is exploited here to discover in which manner the subtasks are prioritized.

For a controller handling constraints both in configuration and operational spaces, some of the most common candidate projection operators are presented in Table 3, covering a very wide range of robotics applications.

Note here that the Gaussian product is computed in configuration space ($\boldsymbol{q}$ and $\boldsymbol{x}$ represent respectively poses in joint space and task space). Eq. (39) describes joint space constraints in a fixed frame. It corresponds to the canonical frame defined by $\boldsymbol{A}_{t,j} = \boldsymbol{I}$ (identity matrix) and $\boldsymbol{b}_{t,j} = \boldsymbol{0}$. Eq. (40) describes absolute position constraints (in operational space), where $\boldsymbol{J}^{\dagger}$ is the Jacobian pseudoinverse used as least-norm inverse kinematics solution. Note that Eq. (40) describes a moving frame, where the task parameters change at each iteration (observation of a changing pose in configuration space). Eq. (41) describes relative position constraints, where the constraint in task space is related to an object described at each time step t by a position $\boldsymbol{b}_t^{\mathcal{O}}$ and an orientation matrix $\boldsymbol{A}_t^{\mathcal{O}}$ in task space. Eq. (42) describes nullspace/priority constraints in joint space, with $\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$ a nullspace projection operator. Eq. (43) describes absolute position nullspace/priority constraints, where the secondary objective is described in task space (for a point in the kinematic chain with corresponding Jacobian $\tilde{\boldsymbol{J}}$). Finally, Eq. (44) describes relative position nullspace/priority constraints.

The above equations can be retrieved without much effort by discretizing (with an Euler approximation) the standard inverse kinematics and nullspace control relations that can be found in most robotics textbooks, see e.g. [5].

Fig. 10 shows an example of constraints in configuration and operational spaces. The task requires the robot to consider, in parallel and in series, constraints in task space (pointing to objects) and in joint space (maintaining a preferred posture). The frame in joint space is defined by Eq. (42), the frames in task space are defined by Eq. (41) for the two objects and by Eq. (40) for the motion in the robot frame, with $\boldsymbol{x}$ encoding Euler angles and $\boldsymbol{J}$ describing the orientation part of the end-effector Jacobian.

Fig. 11 presents a TP-GMM example with task parameters taking the form of nullspace bases. The frames are defined by Equations (41) and (44) with two different combinations of nullspaces and Jacobians corresponding to the left and right arm.

Matlab/GNU Octave implementations of TP-GMM with task parameters in both operational and configuration spaces (including priority constraints) are provided in `demoIK_pointing_TPGMM01.m` and `demoIK_nullspace_TPGMM01.m`.

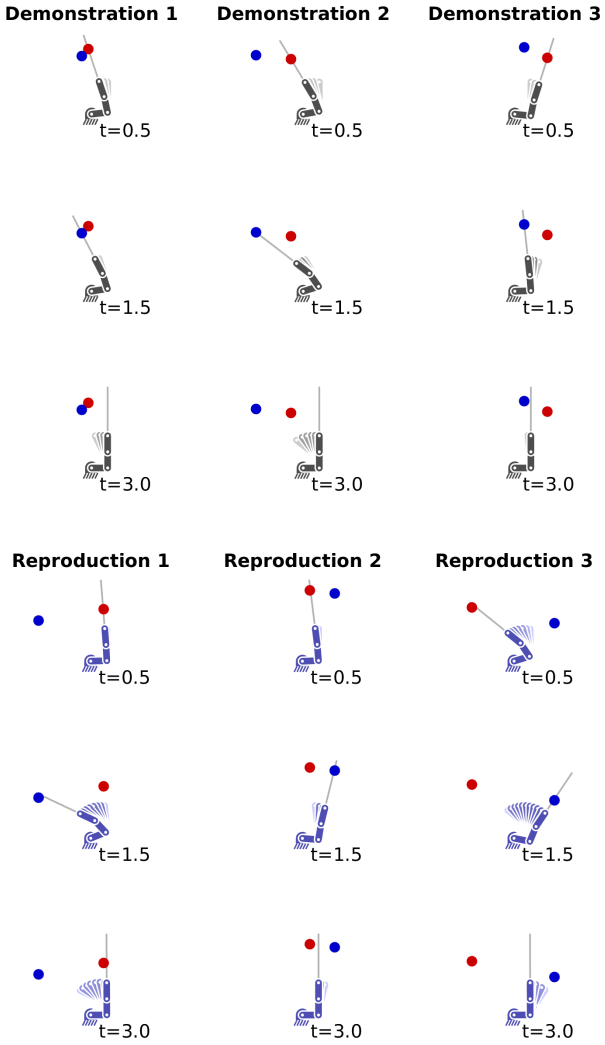


Figure 10: Example of TP-GMM with constraints in both joint space (frame 1) and task space (frames 2-4). The task consists of pointing at two objects (red and blue) and then coming back to a neutral pose. *Left:* The demonstrations show a preferred posture employed to complete the task (first robot link oriented to the right and the other two links oriented upwards). *Right:* Reproduction attempts by synthesizing a motion for the newly encountered situations (new position of blue and red objects), which is achieved through GMR. Each row in the graph shows the configuration at a different time step of the movement. We can see that the generated motion satisfies the demonstrated constraints (pointing sequentially at the two objects while trying to maintain a preferred configuration in joint space). Note here that the motion is generated in an online manner, which allows the robot to handle objects that are moving during the execution of the pointing gestures.

Table 3: Task parameters as candidate projection operators (with affine transformations defined by $\boldsymbol{A}_{t,j}$ and $\boldsymbol{b}_{t,j}$).

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{I} \qquad \boldsymbol{\mu}_i^{(j)} + \boldsymbol{0} \tag{39}$$

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1}) \qquad \boldsymbol{\mu}_i^{(j)} + \boldsymbol{q}_{t-1} - \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{x}_{t-1} \tag{40}$$

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{A}_t^{\mathcal{O}} \qquad \boldsymbol{\mu}_i^{(j)} + \boldsymbol{q}_{t-1} + \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\big[\boldsymbol{b}_t^{\mathcal{O}} - \boldsymbol{x}_{t-1}\big] \tag{41}$$

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{N}(\boldsymbol{q}_{t-1}) \qquad \boldsymbol{\mu}_i^{(j)} + \boldsymbol{J}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{J}(\boldsymbol{q}_{t-1})\boldsymbol{q}_{t-1} \tag{42}$$

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1}) \qquad \boldsymbol{\mu}_i^{(j)} + \boldsymbol{q}_{t-1} - \boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1})\,\boldsymbol{x}_{t-1} \tag{43}$$

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \underbrace{\boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1})\boldsymbol{A}_t^{\mathcal{O}}}_{\boldsymbol{A}_{t,j}} \qquad \boldsymbol{\mu}_i^{(j)} + \underbrace{\boldsymbol{q}_{t-1} + \boldsymbol{N}(\boldsymbol{q}_{t-1})\tilde{\boldsymbol{J}}^{\dagger}(\boldsymbol{q}_{t-1})\big[\boldsymbol{b}_t^{\mathcal{O}} - \boldsymbol{x}_{t-1}\big]}_{\boldsymbol{b}_{t,j}}, \tag{44}$$

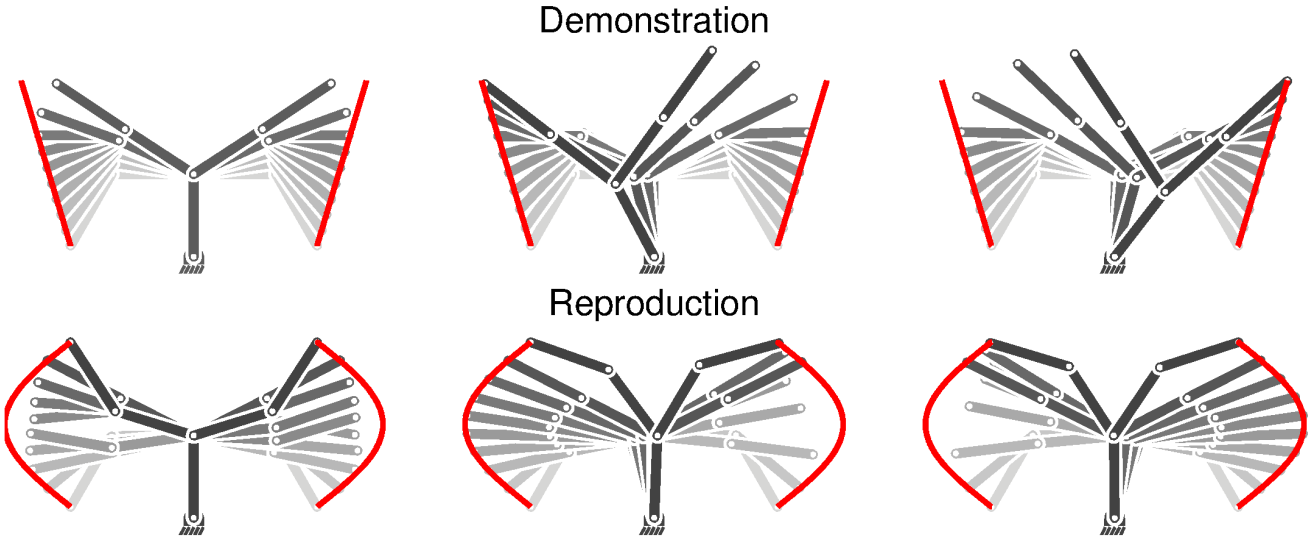
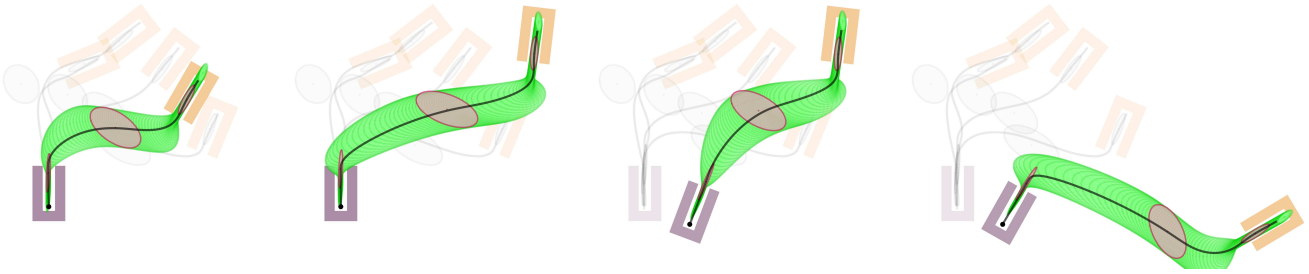
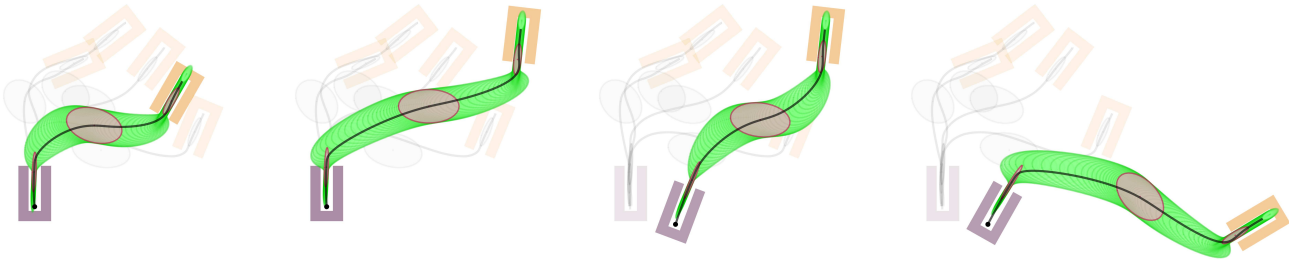


Figure 11: Illustration of the encoding of priority constraints in a TP-GMM. The top row shows 3 demonstrations with a bimanual robot composed of 5 articulations (the color goes from light gray to black to show the evolution of the movement). The task consists of tracking two objects with the left and right end-effectors (the path of the objects are depicted in red). In some parts of the demonstrations, the two objects could not be reached, and the demonstrator either made a compromise (*left graph*), or gave priority to the left or right hand (*middle and right graphs*). The bottom row shows reproduction attempts for new trajectories of the two objects. We can see that, although faced with different situations, the priority constraints are reproduced similarly to the corresponding demonstrations.

Task-parameterized Gaussian mixture model (TP-GMM), see Section 3



Task-parameterized mixture of factor analyzers (TP-MFA) with $d=1$, see Section 4.2



Task-parameterized trajectory GMM (TP-trajGMM), see Section 5.2

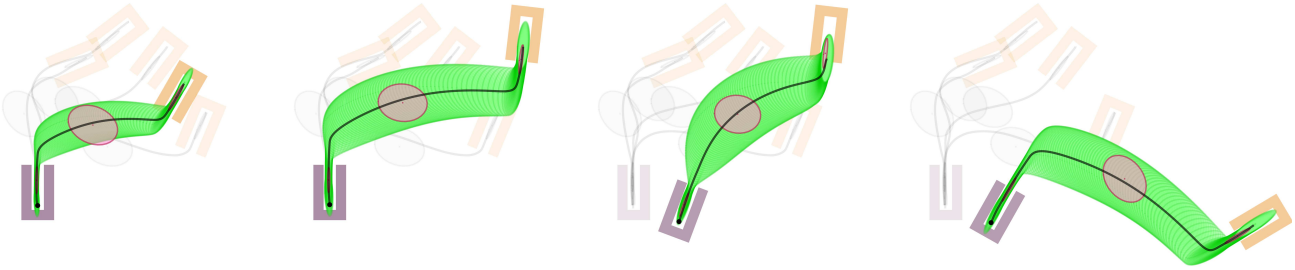


Figure 12: Generalization capability of three model-based task-parameterized approaches. Each row shows the results for a different approach, and each column shows a different situation (with increasing generalization complexity). In each graph, the four demonstrations and the associated adapted model parameters are depicted in semi-transparent colors.

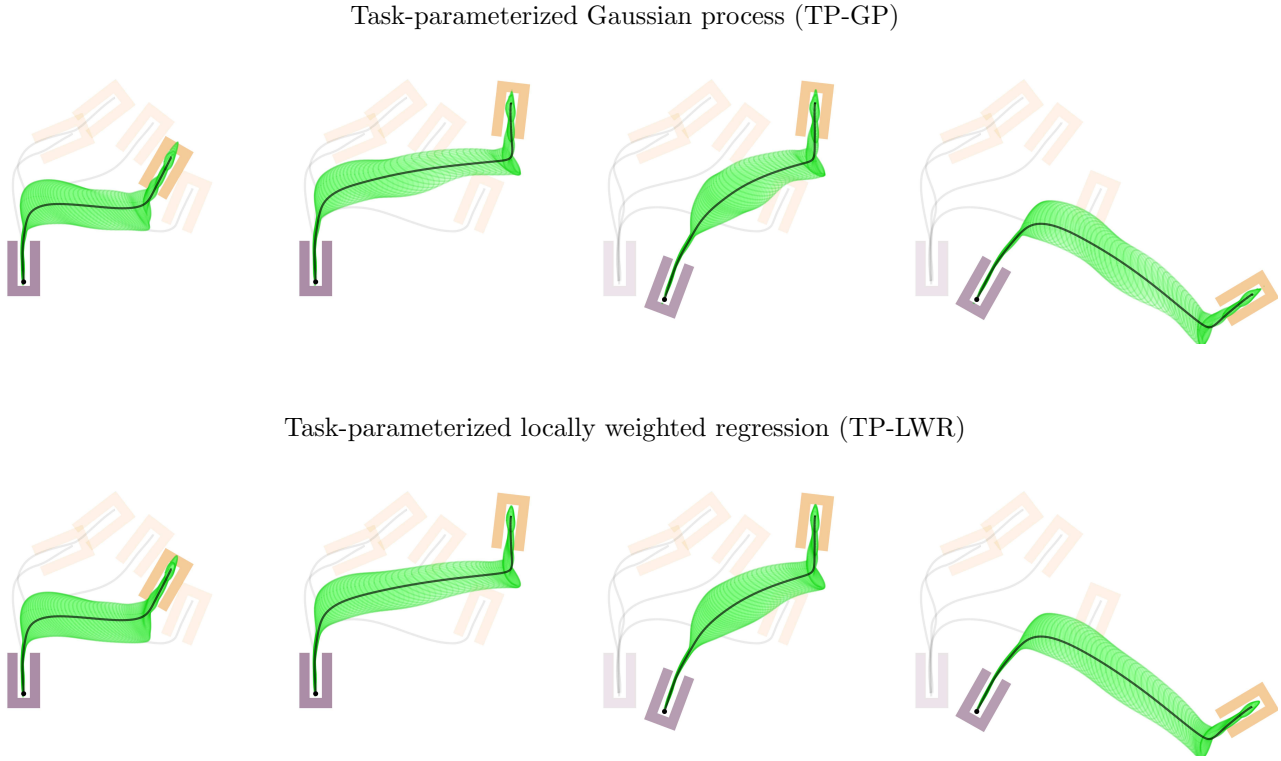


Figure 13: Generalization capability of two data-driven task-parameterized approaches. Each row shows the results for a different approach, and each column shows a different situation (with increasing generalization complexity). In each graph, the four demonstrations and the associated adapted model parameters are depicted in semi-transparent colors.

8 Comparisons with other task-adaptive approaches

This section aims at comparing different techniques to adapt movements to new situations, by using the task adaptation problem depicted in Figures 1 and 2.

Qualitative comparisons for the task-parameterized approaches presented throughout the article are presented in Figures 12 and 13.

Fig. 12 shows that GMM/GMR MFA/GMR and trajectory-GMM could all retrieve suitable trajectories for each of the new situations.

Fig. 13 shows that the proposed task-parameterized approach can also be employed with data-driven encoding strategies, at the expense of a slower reproduction process that depends on the number of demonstrations. For this simple dataset, a kernel-based approach (first row) generated new trajectories that are similar to the trajectories generated by the weighted least-squares approach (second row).

A *Matlab/GNU Octave* implementation of data-driven task-parameterized approach is provided in `demo_TPGP01.m`.

The same task adaptation problem was also tested with two baseline approaches that are described next.

8.1 Gaussian process regression (GPR) with trajectory models

An alternative way of handling task-parameterized movements is to fit a model to each demonstration and associate it with a task-specific feature, goal, style variable or perceptual feedback, see for example [29, 47, 59, 50, 97, 21, 44].

Such approach is typically better suited for task parameters that do not change during the demonstration. It can be achieved in a non-parametric or parametric way, by either encoding the relationships between the raw trajectory variables and the task parameters (first row in Fig. 14), or by encoding the relationships between the trajectory model parameters and the task parameters (second row in Fig. 14, see also Fig. 1).

In this second approach, the output variables $\boldsymbol{\Theta}$ are the model parameters and the query points $\boldsymbol{Q}$ are the task parameters. The reproduction step consists of retrieving new model parameters $\boldsymbol{\Theta}^d$ from new task parameters $\boldsymbol{Q}^d$, which can for example be achieved with *Gaussian process regression* (GPR) [74]. After centering the training data, the joint distribution of the demonstrated and new outputs can be estimated as

$$\begin{bmatrix}\boldsymbol{\Theta}\\ \boldsymbol{\Theta}^d\end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\boldsymbol{K}(\boldsymbol{Q},\boldsymbol{Q})+\sigma^2\boldsymbol{I} & \boldsymbol{K}(\boldsymbol{Q},\boldsymbol{Q}^d)\\ \boldsymbol{K}(\boldsymbol{Q}^d,\boldsymbol{Q}) & \boldsymbol{K}(\boldsymbol{Q}^d,\boldsymbol{Q}^d)\end{bmatrix}\right), \quad (45)$$

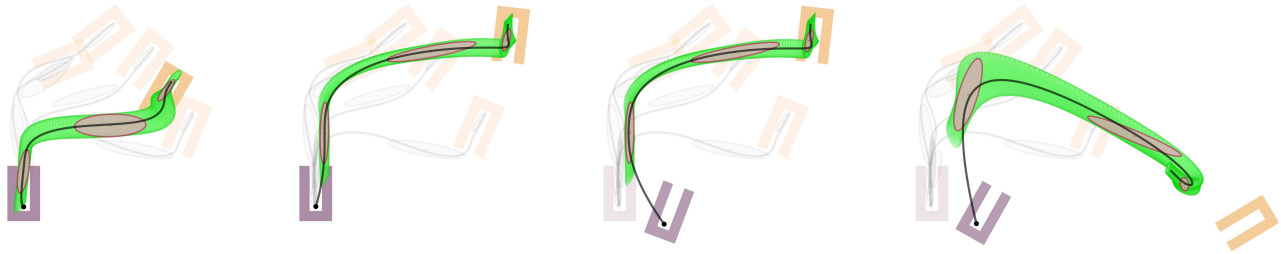
where $\boldsymbol{Q}$ is the concatenation of query points $\boldsymbol{q}_m$, and $\boldsymbol{\Theta}$ the concatenation of outputs $\boldsymbol{\theta}_m$, with $m \in \{1,\dots,M\}$ (M is the number of demonstrations). Squared exponential covariance functions $\boldsymbol{K}$ are considered here.

By using the conditional probability property of normal distributions, the expected outputs $\hat{\boldsymbol{\Theta}}$ associated with the new query points $\boldsymbol{Q}^d$ is given by

$$\hat{\boldsymbol{\Theta}} = \boldsymbol{K}(\boldsymbol{Q}^d,\boldsymbol{Q})\,[\boldsymbol{K}(\boldsymbol{Q},\boldsymbol{Q})+\sigma^2\boldsymbol{I}]^{-1}\boldsymbol{\Theta}, \quad (46)$$

Gaussian process regression with raw trajectory encoding (GPR), see Section 8.1

Gaussian process regression with GMM trajectory encoding (GPR-GMM), see Section 8.1



Parametric Gaussian mixture model (PGMM), see Section 8.2

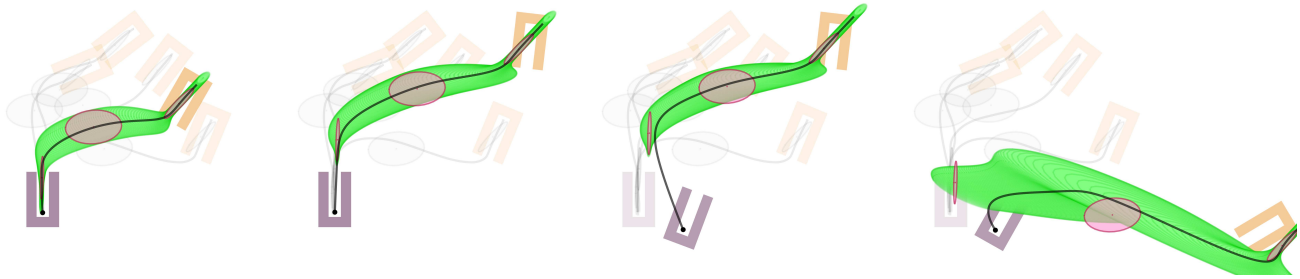


Figure 14: Generalization capability of three alternative approaches to task parameterization. Each row shows the results for a different approach, and each column shows a different situation (with increasing generalization complexity). In each graph, the four demonstrations and the associated adapted model parameters are depicted in semi-transparent colors.

with the covariance of the prediction given by

$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{\Theta}} = \boldsymbol{K}(\boldsymbol{Q}^d,\boldsymbol{Q}^d) - \boldsymbol{K}(\boldsymbol{Q}^d,\boldsymbol{Q})[\boldsymbol{K}(\boldsymbol{Q},\boldsymbol{Q}) + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{K}(\boldsymbol{Q},\boldsymbol{Q}^d). \quad (47)$$

The above formulation can be used with various trajectory models. For a fair comparison, GMM encoding of trajectories was considered, with GMR used to regenerate the trajectories. Thus, a GMM $\boldsymbol{\theta}_m = \{\pi_{i,m}, \boldsymbol{\mu}_{i,m}, \boldsymbol{\Sigma}_{i,m}\}_{i=1}^K$ is fit to each demonstration $\boldsymbol{\xi}_m$, with associate query point $\boldsymbol{q}_m = \{\boldsymbol{A}_{m,j}, \boldsymbol{b}_{m,j}\}_{j=1}^P$ describing the demonstration context. After all demonstrations are collected, $[\boldsymbol{K}(\boldsymbol{Q},\boldsymbol{Q}) + \sigma^2\boldsymbol{I}]^{-1}\boldsymbol{\Theta}$ in Equations (45)-(46) can be precomputed.

During reproduction, a new query point $\boldsymbol{Q}^d$ with $\boldsymbol{Q}^d = \{\boldsymbol{A}_j, \boldsymbol{b}_j\}_{j=1}^P$ is used to retrieve a new GMM using Eq. (46), which is then used to retrieve a new trajectory through GMR.

The first and second row of Fig. 14 show generalization results for the use of GPR with a non-parametric (first row) and parametric (second row) encoding of the data. Although GPR can easily handle situations within the range of the demonstrations, its generalization capability can degrade if the query points are too far from the demonstrations (it collapses to an average of the models), which is confirmed by the results of the experiment.

A *Matlab/GNU Octave* implementation of GPR with trajectory models is provided in `demo_GPR01.m`.

8.2 Parametric HMM/GMM (PHMM/PGMM)

Another approach to handle task-parameterized movements is to encode all demonstrations in a single mixture model, where each cluster in the mixture learns the relations between the task parameters and the motion. The *parametric hidden Markov model* (PHMM) is a representative approach in this category. It was originally introduced for recognition and prediction of gestures [102], and extended in robotics to movement generation [105, 51]. We will refer to a *parametric Gaussian mixture model* (PGMM) when the transition and initial state probabilities are not taken into account in the likelihood estimation.[7]

The original model modulates in each cluster the center of a Gaussian distribution through an affine relationship with the task parameters. This is achieved by concatenating the task parameters in a vector $\boldsymbol{Q}_t$ as in the above, and defining the centers of the Gaussians in a task-adaptive manner with

$$\boldsymbol{\mu}_{t,i} = \boldsymbol{Z}_i \left[\boldsymbol{Q}_t^\top,\ 1\right]^\top, \quad (48)$$

where $\tilde{\boldsymbol{Z}}_i$ describe the model parameters to be estimated. In the case of affine relationships, this can be done with EM, see Appendix D for details. The other parameters of the model are estimated as the standard GMM formulation.

After having trained the model, each new set of task parameters concatenated in $\boldsymbol{Q}_t$ will provide new Gaussian centers $\boldsymbol{\mu}_{t,i}$ in the GMM by using Eq. (48), where GMR can then be used to retrieve a new trajectory.

The last row of Fig. 14 shows generalization results with PGMM. The main drawback of this model is that only the centers of the Gaussians are adapted to the current situation. The covariances are estimated as constant matrices $\boldsymbol{\Sigma}_i$, estimated with the standard EM procedure for GMM. This limitation is confirmed in the experiment, where EM often converged to local optima that were unable to extract the underlying structures of the task for the encoding of continuous movements.

A *Matlab/GNU Octave* implementation of parametric GMM is provided in `demo_stdPGMM01.m`. Figures 12-14 were generated by using the *Matlab/GNU Octave* codes `benchmark_DS_*.m`, with `*` to be replaced by the corresponding method.

9 Discussion and future work

Recent service robots are provided with numerous and/or high resolution sensors and actuators. This increase of dimensionality is problematic in applications where the sample size remains bounded by the cost of data acquisition. There is a non-negligible number of applications that would require models capable of targeting *wide-ranging data*. Namely, models that could start learning from a small number of demonstrations, while still being able to continue learning once more data become available. Robot learning from demonstration is one such field, whose learning challenge often requires the design of appropriate domain-specific priors to ensure that generalization can be achieved from small training sets.

We showed throughout this article that an efficient and versatile prior knowledge for task adaptation is to consider that the task parameters describing the current situation (body and workspace configuration that the robot encounters) can be represented as affine transformations (including frames of reference, coordinate systems or projections).

The above prior requires the experimenter to provide the robot with a set of candidate frames that could potentially be relevant for the task. We showed that the representation as affine transformations has a simple interpretation, can be easily implemented, and remains valid for a wide range of skills that a service robot can encounter. It was shown in [4] that when frames are missing during reproduction (e.g., when occlusions occur or when frames are collected at different rates), the system is still able to reproduce an appropriate behavior for the current circumstance.

A limitation of the current TP-GMM approach is that it requires the experimenter to provide an initial set of frames that will act as candidate projections/transformations of the data that might potentially be relevant for the task. The number of frames can be overspecified by the experimenter (e.g., by providing an exhaustive list), but it comes at the expense of requiring more demonstrations to obtain sufficient statistics to discard the frames that have no role in the task. The problem is not different from the problem of selecting the variables

[7] Note here that the term *parametric* in PGMM/PHMM (referring to task parameters) is ambiguous because a standard GMM can also be described as being parametric (referring to model parameters).

that will form the feature vectors fed to a learning process. The only difference lies in the selection of frames in the form of affine transformations that are most often associated with easily interpretable behaviors.

In practice, the experimenter selects objects and locations in the robot kinematic chain that might be relevant for the task, which are typically the end-effectors of the robot, where tools, grippers or parts in contact with the environment are mounted, see also discussion in [45].[8] The issue of predefining an initial set of frames is not restrictive when the number of frames remains reasonably low (e.g., when they come from a set of predefined objects tracked with visual markers in a lab setting). However, for perception in unconstrained environment, the number of frames could grow quickly (e.g., detection of phantom objects), while the number of demonstrations remains low. Further work is thus required to detect redundant frames or remove irrelevant frames, as well as to automatically determine in which manner the frames are coordinated with each other and locally contribute to the achievement of the task. A promising route for further investigation is to exploit the recent developments in multilinear algebra and tensor analysis [85, 48] that exploit the multivariate structure of the data for statistical analysis and compression, without transforming it to a matrix form (by processing data jointly in spatial and spectral ways, instead of flattening the higher-order tensor dataset).

In service robotics, movements often need to be expressed simultaneously in multiple coordinate systems, and are stored as multidimensional arrays (tensor-variate data). Multilinear algebra could thus provide a principled method to simultaneously extract *eigenframes*, *eigenposes* and *eigentrajectories*. Multiway analysis of tensor-variate data offers a rich set of data decomposition techniques, whose advantage has been demonstrated in computer imaging fields such as face processing [98], video analysis [107], geoscience [75] or neuroimaging [8], but which remains an uncharted territory in robotics and motor skills learning.

Another open route for further investigation concerns the use of a richer set of task parameters. A wide range of motor skills could potentially be adapted to this framework, by exploiting the functional nature of task parameters to build models that learn the local structure of the task from a low number of demonstrations. Indeed, most task parameterization in robot control can be related to some form of frames of reference, coordinate systems, projections or basis functions, where the involvement of the frames can change during the execution of the task, with transformations represented as locally linear projection operators (e.g., Jacobians for inverse kinematics, kernel matrix for nullspace projections, etc.). A wider range of robot skills could be defined in such way, see e.g. the possible tasks described in §6.2.1 of [5].

The potential applications are diverse, with an objective in line with the original purpose of motor primitives to be composed together serially or in parallel [28]. TP-GMM could potentially be employed as a tool to revisit existing robotics techniques in a probabilistic form. This includes the consideration of soft constraints in both configuration and operational spaces, where the frames would correspond to different subspace projections of the same movement, with the extracted regularities employed to learn bimanual tasks or whole-body movements.

One of the important next challenge in robot learning will be to extend the concept of movement primitives to a broader range of behaviors including impedance, reaction and collaboration primitives.

[8] Full end-effector poses or decoupled position and orientation can be considered here.

10 Conclusion

In service robotics, movements often need to be modulated by external parameters describing the current situation (body and workspace configuration that the robot encounters). This tutorial showed that in many cases, the task parameters can be represented as affine transformations. Based on this prior assumption, a task-parameterized model was presented by exploiting this structure to learn a skill from a small number of demonstrations.

The proposed approach was implemented and tested with various statistical encoding strategies, including standard mixture models, kernel approaches and subspace clustering methods. It was shown that a wide variety of problems in robotics can be reinterpreted by introducing such relation between the task parameters and the model parameters. The approach was demonstrated in a series of control and planning problems in operational and configuration spaces. Each section of the article was accompanied with source codes to help the practitioners study, test and extend the proposed approach.

Appendices

A Expectation-Maximization for TP-GMM parameters estimation

In order to estimate the parameters of a TP-GMM, the following two steps are repeated until convergence.
E-step:

$$h_{t,i} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}\Big(\boldsymbol{X}_t^{(j)} \Big| \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\Big)}{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{P} \mathcal{N}\Big(\boldsymbol{X}_t^{(j)} \Big| \boldsymbol{\mu}_k^{(j)}, \boldsymbol{\Sigma}_k^{(j)}\Big)}. \quad (49)$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}}{N}, \quad (50)$$

$$\boldsymbol{\mu}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}\, \boldsymbol{X}_t^{(j)}}{\sum_{t=1}^{N} h_{t,i}}, \quad (51)$$

$$\boldsymbol{\Sigma}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i} \Big(\boldsymbol{X}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\Big)\Big(\boldsymbol{X}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\Big)^\top}{\sum_{t=1}^{N} h_{t,i}}. \quad (52)$$

In practice, it is recommended to start EM from a coarse estimate of the parameters. For example, based on an equal split in time of motion segments, based on a geometric segmentation with k-means [58], based on moments or

spectral approaches with circular covariances [84, 52, 42], or based on an iterative clustering algorithm [82].

Model selection (i.e., determining the number of Gaussians in the GMM) is compatible with the techniques employed in standard GMM, such as the use of a *Bayesian information criterion* [81], *Dirichlet process* [73, 22, 64, 49], *iterative pairwise replacement* [82], *spectral clustering* [68, 84, 52], or based on segmentation points [55]. Model selection in mixture modeling shares a similar core challenge as that of data-driven sparse kernel regression techniques, which requires to find the right bandwidth parameters to select a subset of existing/new datapoints that are the most representatives of the dataset.

B Expectation-Maximization for TP-MFA and TP-MPPCA parameters estimation

In TP-MFA, the generative model for the j-th frame and i-th mixture component assumes that a D-dimension random vector $\boldsymbol{X}^{(j)}$ is modeled using a d-dimension vector of latent (unobserved) factors $\boldsymbol{z}^{(j)}$

$$\boldsymbol{X}^{(j)} = \boldsymbol{\Lambda}_i^{(j)}\boldsymbol{z}^{(j)} + \boldsymbol{\mu}_i^{(j)} + \boldsymbol{\epsilon}_i^{(j)}, \tag{53}$$

where $\boldsymbol{\mu}_i^{(j)} \in \mathbb{R}^D$ is the mean vector of the i-th factor analyzer, $\boldsymbol{z}^{(j)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ (the factors are assumed to be distributed according to a zero-mean normal with unit variance), and $\boldsymbol{\epsilon}_i^{(j)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Psi}_i^{(j)})$ is a centered normal noise with diagonal covariance $\boldsymbol{\Psi}_i^{(j)}$.

This diagonality is a key assumption in factor analysis. Namely, the observed variables are independent given the factors, and the goal of TP-MFA is to best model the covariance structure of $\boldsymbol{X}^{(j)}$. It follows from this model that the marginal distribution of $\boldsymbol{X}^{(j)}$ for the i-th component is

$$\boldsymbol{X}^{(j)} \sim \mathcal{N}\left(\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Lambda}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)^\top} + \boldsymbol{\Psi}_i^{(j)}\right), \tag{54}$$

and the joint distribution of $\boldsymbol{X}^{(j)}$ and $\boldsymbol{z}^{(j)}$ is

$$\begin{bmatrix}\boldsymbol{X}^{(j)} \\ \boldsymbol{z}^{(j)}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}_i^{(j)} \\ \boldsymbol{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Lambda}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)^\top} + \boldsymbol{\Psi}_i^{(j)} & \boldsymbol{\Lambda}_i \\ \boldsymbol{\Lambda}_i^{(j)^\top} & \boldsymbol{I}\end{bmatrix}\right). \tag{55}$$

The above can be used to show that the d factors are informative projections of the data, which can be computed by Gaussian conditioning, corresponding to the affine projection

$$\mathbb{E}\left(\boldsymbol{z}^{(j)}|\boldsymbol{X}^{(j)}\right) = \overbrace{\boldsymbol{\Lambda}_i^{(j)^\top}\left(\boldsymbol{\Lambda}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)^\top} + \boldsymbol{\Psi}_i^{(j)}\right)^{-1}}^{\boldsymbol{B}_i^{(j)}}\left(\boldsymbol{\mu}_i^{(j)} - \boldsymbol{X}^{(j)}\right). \tag{56}$$

As highlighted by [32], the same process can be used to estimate the second moment of the factors $\mathbb{E}\left(\boldsymbol{z}^{(j)}\boldsymbol{z}^{(j)^\top}|\boldsymbol{X}^{(j)}\right)$, which provides a measure of uncertainty in the factors that has no analogue in PCA. This relation can be exploited to derive an EM algorithm (see for example [32] or [61]) to train a TP-MFA model of K components with parameters $\left\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Lambda}_i^{(j)}, \boldsymbol{\Psi}_i^{(j)}\}_{j=1}^P\right\}_{i=1}^K$, yielding an EM parameters estimation strategy.

The following two steps are repeated until convergence.

E-step:

$$h_{t,i} = \frac{\pi_i \prod_{j=1}^{P} \mathcal{N}\left(\boldsymbol{X}_t^{(j)} \middle| \boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Lambda}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)^\top} + \boldsymbol{\Psi}_i^{(j)}\right)}{\sum_{k=1}^{K} \pi_k \prod_{j=1}^{P} \mathcal{N}\left(\boldsymbol{X}_t^{(j)} \middle| \boldsymbol{\mu}_k^{(j)}, \boldsymbol{\Lambda}_k^{(j)}\boldsymbol{\Lambda}_k^{(j)^\top} + \boldsymbol{\Psi}_k^{(j)}\right)}. \tag{57}$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}}{N}, \tag{58}$$

$$\boldsymbol{\mu}_i^{(j)} \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}\boldsymbol{X}_t^{(j)}}{\sum_{t=1}^{N} h_{t,i}}, \tag{59}$$

$$\boldsymbol{\Lambda}_i^{(j)} \leftarrow \boldsymbol{S}_i^{(j)}\boldsymbol{B}_i^{(j)^\top}\left(\boldsymbol{I} - \boldsymbol{B}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)} + \boldsymbol{B}_i^{(j)}\boldsymbol{S}_i^{(j)}\boldsymbol{B}_i^{(j)^\top}\right)^{-1}, \tag{60}$$

$$\boldsymbol{\Psi}_i^{(j)} \leftarrow \mathrm{diag}\Big(\mathrm{diag}\big(\boldsymbol{S}_i^{(j)} - \boldsymbol{\Lambda}_i^{(j)}\boldsymbol{B}_i^{(j)}\boldsymbol{S}_i^{(j)}\big)\Big), \tag{61}$$

computed with the help of the intermediary variables

$$\boldsymbol{S}_i^{(j)} = \frac{\sum_{t=1}^{N} h_{t,i}\left(\boldsymbol{X}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\right)\left(\boldsymbol{X}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\right)^\top}{\sum_{t=1}^{N} h_{t,i}}, \tag{62}$$

$$\boldsymbol{B}_i^{(j)} = \boldsymbol{\Lambda}_i^{(j)^\top}\left(\boldsymbol{\Lambda}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)^\top} + \boldsymbol{\Psi}_i^{(j)}\right)^{-1}. \tag{63}$$

Alternatively, an update step simultaneously computing $\boldsymbol{\mu}_i^{(j)}$ and $\boldsymbol{\Lambda}_i^{(j)}$ can be derived, see [32] for details.

Similarly, the M-step in TP-MPPCA is given by

$$\tilde{\boldsymbol{\Lambda}}_i^{(j)} \leftarrow \boldsymbol{S}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)}\left(\boldsymbol{I}{\sigma_i^{(j)}}^2 + {\boldsymbol{M}_i^{(j)}}^{-1}\boldsymbol{\Lambda}_i^{(j)^\top}\boldsymbol{S}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)}\right)^{-1}, \tag{64}$$

$$\boldsymbol{\Psi}_i^{(j)} \leftarrow \boldsymbol{I}{\sigma_i^{(j)}}^2, \tag{65}$$

computed with the help of the intermediary variables

$$\boldsymbol{S}_i^{(j)} = \frac{\sum_{t=1}^{N} h_{t,i}\left(\boldsymbol{\xi}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\right)\left(\boldsymbol{\xi}_t^{(j)} - \boldsymbol{\mu}_i^{(j)}\right)^\top}{\sum_{t=1}^{N} h_{t,i}}, \tag{66}$$

$$\boldsymbol{M}_i^{(j)} = \boldsymbol{\Lambda}_i^{(j)^\top}\boldsymbol{\Lambda}_i^{(j)} + \boldsymbol{I}{\sigma_i^{(j)}}^2, \tag{67}$$

$${\sigma_i^{(j)}}^2 = \frac{1}{D}\mathrm{tr}\Big(\boldsymbol{S}_i^{(j)} - \boldsymbol{S}_i^{(j)}\boldsymbol{\Lambda}_i^{(j)}{\boldsymbol{M}_i^{(j)}}^{-1}\tilde{\boldsymbol{\Lambda}}_i^{(j)^\top}\Big), \tag{68}$$

where $\boldsymbol{\Lambda}_i^{(j)}$ is replaced by $\tilde{\boldsymbol{\Lambda}}_i^{(j)}$ at each iteration, see [93] for details.

C Gaussian mixture regression approximated by a single normal distribution

Let us consider a datapoint $\boldsymbol{\xi}_t$ distributed as in Eq. (6), with $\mathcal{P}(\boldsymbol{\xi}_t) = \mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}}, \boldsymbol{\xi}_t^{\mathcal{O}})$ the joint distribution describing the data. The conditional probability of an output given an input is

$$\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \frac{\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}}, \boldsymbol{\xi}_t^{\mathcal{O}})}{\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}})} = \frac{\sum_{i=1}^{K}\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}}, \boldsymbol{\xi}_t^{\mathcal{O}}|z_i)\mathcal{P}(z_i)}{\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}})}, \tag{69}$$

where z_i represents the i-th component of the GMM. Namely,

$$\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \sum_{i=1}^{K} \mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}, z_i) \frac{\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}}|z_i)\mathcal{P}(z_i)}{\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}})} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\, \mathcal{N}\Big(\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}), \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big). \tag{70}$$

The conditional mean can be computed as

$$\hat{\boldsymbol{\mu}}_t^{\mathcal{O}} = \mathbb{E}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \int \boldsymbol{\xi}_t^{\mathcal{O}}\, \mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}})\, d\boldsymbol{\xi}_t^{\mathcal{O}} = \int \boldsymbol{\xi}_t^{\mathcal{O}} \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\, \mathcal{N}\Big(\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}), \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big)\, d\boldsymbol{\xi}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\, \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}). \tag{71}$$

In order to evaluate the covariance, we calculate

$$\mathrm{cov}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \mathbb{E}(\boldsymbol{\xi}_t^{\mathcal{O}}\boldsymbol{\xi}_t^{\mathcal{O}\top}|\boldsymbol{\xi}_t^{\mathcal{I}}) - \mathbb{E}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}})\mathbb{E}(\boldsymbol{\xi}_t^{\mathcal{O}\top}|\boldsymbol{\xi}_t^{\mathcal{I}}). \tag{72}$$

We have that

$$\mathbb{E}(\boldsymbol{\xi}_t^{\mathcal{O}}\boldsymbol{\xi}_t^{\mathcal{O}\top}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \int \boldsymbol{\xi}_t^{\mathcal{O}}\boldsymbol{\xi}_t^{\mathcal{O}\top}\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}})\, d\boldsymbol{\xi}_t^{\mathcal{O}} = \int \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\, \boldsymbol{\xi}_t^{\mathcal{O}}\boldsymbol{\xi}_t^{\mathcal{O}\top}\mathcal{N}\Big(\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}), \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big)\, d\boldsymbol{\xi}_t^{\mathcal{O}} = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}}) \int \boldsymbol{\xi}_t^{\mathcal{O}}\boldsymbol{\xi}_t^{\mathcal{O}\top}\mathcal{N}\Big(\hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}}), \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big)\, d\boldsymbol{\xi}_t^{\mathcal{O}}. \tag{73}$$

By using Eq. (72) with a Gaussian distribution, we obtain

$$\mathbb{E}(\boldsymbol{\xi}_t^{\mathcal{O}}\boldsymbol{\xi}_t^{\mathcal{O}\top}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} + \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\, \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}})\, \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}})^{\top}. \tag{74}$$

Combining (72) with (74) we finally have that (see also [90])

$$\hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}} = \mathrm{cov}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \sum_{i=1}^{K} h_i(\boldsymbol{\xi}_t^{\mathcal{I}})\Big(\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} + \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}})\, \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}(\boldsymbol{\xi}_t^{\mathcal{I}})^{\top}\Big) - \hat{\boldsymbol{\mu}}_t^{\mathcal{O}}\hat{\boldsymbol{\mu}}_t^{\mathcal{O}\top}. \tag{75}$$

D Expectation-Maximization for parametric GMM parameters estimation

The following two steps are repeated until convergence, see [102] for details.

E-step:

$$h_{t,i} = \frac{\pi_i \mathcal{N}\big(\boldsymbol{\xi}_t\big|\, \boldsymbol{\mu}_{t,i}, \boldsymbol{\Sigma}_i\big)}{\sum_{k=1}^{K} \pi_k \mathcal{N}\big(\boldsymbol{\xi}_t\big|\, \boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_k\big)}. \tag{76}$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}}{N}, \tag{77}$$

$$\boldsymbol{Z}_i \leftarrow \left(\sum_{t=1}^{N} h_{t,i}\, \boldsymbol{\xi}_t\big[\boldsymbol{Q}_t^{\top},\ 1\big]\right)\left(\sum_{t=1}^{N} h_{t,i}\big[\boldsymbol{Q}_t^{\top},\ 1\big]^{\top}\big[\boldsymbol{Q}_t^{\top},\ 1\big]\right)^{-1}, \tag{78}$$

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}\, (\boldsymbol{\xi}_t - \boldsymbol{\mu}_{t,i})(\boldsymbol{\xi}_t - \boldsymbol{\mu}_{t,i})^{\top}}{\sum_{t=1}^{N} h_{t,i}}, \tag{79}$$

$$\text{where} \quad \boldsymbol{\mu}_{t,i} = \boldsymbol{Z}_i\, \big[\boldsymbol{Q}_t^{\top},\ 1\big]^{\top}. \tag{80}$$

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. Intl Conf. on Machine Learning (ICML)*, 2004.

[2] B. Akgun and A. Thomaz. Simultaneously learning actions and goals from demonstration. *Autonomous Robots*, pages 1–17, 2015.

[3] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Action, state and effect metrics for robot imitation. In *Proc. IEEE Intl Symp. on Robot and Human Interactive Communication (Ro-Man)*, pages 232–237, Hatfield, UK, September 2006.

[4] T. Alizadeh, S. Calinon, and D. G. Caldwell. Learning from demonstrations with partially observable task parameters. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3309–3314, Hong Kong, China, May-June 2014.

[5] G. Antonelli. *Underwater Robots*. Springer International Publishing, 2014. 3rd Edition.

[6] K. J. Astrom and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, USA, 2008.

[7] J. Baek, G. J. McLachlan, and L. K. Flack. Mixtures of factor analyzers with common factor loadings: Applications to the clustering and visualization of high-dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(7):1298–1309, 2010.

[8] P. J. Basser and S. Pajevic. A normal distribution for tensor-valued random variables: applications to diffusion tensor MRI. *IEEE Trans. on Medical Imaging*, 22(7):785–794, July 2003.

[9] Y. Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[10] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for linear and hybrid systems*. Cambridge University Press, 2015. In preparation.

[11] C. Bouveyron and C. Brunet. Model-based clustering of high-dimensional data: A review. *Computational Statistics and Data Analysis*, 71:52–78, March 2014.

[12] C. Bouveyron, S. Girard, and C. Schmid. High-dimensional data clustering. *Computational Statistics and Data Analysis*, 52(1):502–519, 2007.

[13] M. Brand and A. Hertzmann. Style machines. In *Proc. ACM Intl Conf. on Computer graphics and Interactive Techniques (SIGGRAPH)*, pages 183–192, New Orleans, Louisiana, USA, July 2000.

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

[15] S. Calinon and A. G. Billard. Statistical learning by imitation of competing constraints in joint space and task space. *Advanced Robotics*, 23(15):2059–2076, 2009.

[16] S. Calinon, D. Bruno, and D. G. Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3339–3344, Hong Kong, China, May-June 2014.

[17] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. 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, June 2010.

[18] S. Calinon, F. Guenter, and A. G. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Trans. on Systems, Man and Cybernetics, Part B*, 37(2):286–298, 2007.

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

[20] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, pages 323–329, Osaka, Japan, 2012.

[21] C. L. Campbell, R. A. Peters, R. E. Bodenheimer, W. J. Bluethmann, E. Huber, and R. O. Ambrose. Superpositioning of behaviors learned through teleoperation. *IEEE Trans. on Robotics*, 22(1):79–91, 2006.

[22] S. P. Chatzis, D. Korkinof, and Y. Demiris. A nonparametric Bayesian approach toward robot learning by demonstration. *Robotics and Autonomous Systems*, 60(6):789–802, 2012.

[23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38, 1977.

[24] A. Doerr, N. Ratliff, J. Bohg, M. Toussaint, and S. Schaal. Direct loss minimization inverse optimal control. In *Proc. Robotics: Science and Systems (R:SS)*, pages 1–9, Rome, Italy, July 2015.

[25] S. Dong and B. Williams. Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. *Intl Journal of Social Robotics*, 4(4):357–368, 2012.

[26] M. Field, D. Stirling, Z. Pan, and F. Naghdy. Learning trajectories for robot programing by demonstration using a coordinated mixture of factor analyzers. *IEEE Trans. on Cybernetics*, 2015.

[27] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):381–396, March 2002.

[28] T. Flash and B. Hochner. Motor primitives in vertebrates and invertebrates. *Current opinion in neurobiology*, 15(6):660–666, 2005.

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

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

[31] M. J. F. Gales. Semi-tied covariance matrices for hidden markov models. *IEEE Trans. on Speech and Audio Processing*, 7(3):272–281, 1999.

[32] Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical report, University of Toronto, 1997.

[33] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 6, pages 120–127, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers, Inc.

[34] N. Greggio, A. Bernardino, P. Dario, and J. Santos-Victor. Efficient greedy estimation of mixture models through a binary tree search. *Robotics and Autonomous Systems*, 62(10):1440–1452, 2014.

[35] D. B. Grimes, R. Chalodhorn, and R. P. N. Rao. Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In *Proc. Robotics: Science and Systems (R:SS)*, pages 1–8, 2006.

[36] R. Gross and J. Shi. The CMU motion of body (MoBo) database. Technical Report CMU-RI-TR-01-18, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, June 2001.

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

[38] M. Hersch, F. Guenter, S. Calinon, and A. G. Billard. Learning dynamical system modulation for constrained reaching tasks. In *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, pages 444–449, Genova, Italy, December 2006.

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

[40] G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007.

[41] N. Hogan and D. Sternad. Dynamic primitives of motor behavior. *Biological Cybernetics*, 106(11–12):727–739, 2012.

[42] D. Hsu and S. M. Kakade. Learning mixtures of spherical Gaussians: Moment methods and spectral decompositions. In *Conf. on Innovations in Theoretical Computer Science*, pages 11–20, 2013.

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

[44] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura. Embodied symbol emergence based on mimesis theory. *Intl Journal of Robotic Research*, 23(4-5):363–377, 2004.

[45] N. Jetchev and M. Toussaint. Discovering relevant task spaces using inverse feedback control. *Autonomous Robots*, 37(2):169–189, 2014.

[46] S. M. Khansari-Zadeh and A. Billard. Learning stable non-linear dynamical systems with Gaussian mixture models. *IEEE Trans. on Robotics*, 27(5):943–957, 2011.

[47] J. Kober, A. Wilhelm, E. Oztop, and J. Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, April 2012.

[48] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[49] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg. Unsupervised surgical task segmentation with milestone learning. In *Proc. Intl Symp. on Robotics Research (ISRR)*, 2015.

[50] K. Kronander, M. S. M. Khansari-Zadeh, and A. Billard. Learning to control planar hitting motions in a minigolf-like task. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 710–717, 2011.

[51] V. Krueger, D. L. Herzog, S. Baby, A. Ude, and D. Kragic. Learning actions from observations: Primitive-based modeling and grammar. *IEEE Robotics and Automation Magazine*, 17(2):30–43, 2010.

[52] B. Kulis and M. I. Jordan. Revisiting k-means: New algorithms via Bayesian nonparametrics. In *Proc. Intl Conf. on Machine Learning (ICML)*, 2012.

[53] M. L. Latash, J. P. Scholz, and G. Schoener. Motor control strategies revealed in the structure of motor variability. *Exerc. Sport Sci. Rev.*, 30(1):26–31, 2002.

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

[55] S. H. Lee, I. H. Suh, S. Calinon, and R. Johansson. Autonomous framework for segmenting robot trajectories of manipulation task. *Autonomous Robots*, 38(2):107–141, February 2015.

[56] S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills with guided policy search. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 156–163, May 2015.

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

[58] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on mathematical statistics and probability*, pages 281–297, 1967.

[59] T. Matsubara, S.-H. Hyon, and J. Morimoto. Learning parametric dynamic movement primitives from multiple demonstrations. *Neural Networks*, 24(5):493–500, June 2011.

[60] G. J. McLachlan, D. Peel, and R. W. Bean. Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics and Data Analysis*, 41(3-4):379–388, 2003.

[61] P. D. McNicholas and T. B. Murphy. Parsimonious Gaussian mixture models. *Statistics and Computing*, 18(3):285–296, September 2008.

[62] J. R. Medina, D. Lee, and S. Hirche. Risk-sensitive optimal feedback control for haptic assistance. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 1025–1031, May 2012.

[63] S. Miller, M. Fritz, T. Darrell, and P. Abbeel. Parametrized shape models for clothing. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 4861–4868, May 2011.

[64] T. M. Moldovan, S. Levine, M. I. Jordan, and P. Abbeel. Optimism-driven exploration for nonlinear systems. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3239–3246, Seattle, WA, USA, May 2015.

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

[66] F. A. Mussa-Ivaldi. From basis functions to basis fields: vector field approximation from sparse data. *Biological Cybernetics*, 67(6):479–489, 1992.

[67] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. MIT Press, Cambridge, MA, USA, 1999.

[68] A. Ng, M. Jordan, and Y. Weiss. On Spectral Clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 849–856. MIT Press, 2001.

[69] D. Nguyen-Tuong and J. Peters. Local Gaussian process regression for real-time model-based robot control. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 380–385, 2008.

[70] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

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

[72] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:2:257–285, February 1989.

[73] C. E. Rasmussen. The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 554–560. MIT Press, 2000.

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

[75] N. Renard, S. Bourennane, and J. Blanc-Talon. Denoising and dimensionality reduction using multilinear tools for hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 5(2):138–142, April 2008.

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

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

[78] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

[79] S. Schaal, P. Mohajerian, and A. J. Ijspeert. Dynamics systems vs. optimal control: a unifying view. *Progress in Brain Research*, 165:425–445, 2007.

[80] J. P. Scholz and G. Schoener. The uncontrolled manifold concept: identifying control variables for a functional task. *Experimental Brain Research*, 126(3):289–306, 1999.

[81] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[82] D. W. Scott and W. F. Szewczyk. From kernels to mixtures. *Technometrics*, 43(3):323–335, 2001.

[83] J. A. Scott Kelso. Synergies: Atoms of brain and behavior. *Progress in Motor Control*, pages 83–91, 2009.

[84] T. Shi, M. Belkin, and B. Yu. Data spectroscopy: eigenspace of convolution operators and clustering. *The Annals of Statistics*, 37(6B):3960–3984, 2009.

[85] M. Signoretto, R. Van de Plas, B. De Moor, and J. A. K. Suykens. Tensor versus matrix completion: A comparison with application to spectral data. *IEEE Signal Processing Letters*, 18(7):403–406, July 2011.

[86] D. Sternad, S.-W. Park, H. Mueller, and N. Hogan. Coordinate dependence of variability analysis. *PLoS Computational Biology*, 6(4):1–16, 04 2010.

[87] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986.

[88] F. Stulp and O. Sigaud. Many regression algorithms, one unified model - a review. *Neural Networks*, 69:60–79, September 2015.

[89] K. Sugiura, N. Iwahashi, H. Kashioka, and S. Nakamura. Learning, generation, and recognition of motions by reference-point-dependent probabilistic models. *Advanced Robotics*, 25(5), 2011.

[90] H. G. Sung. *Gaussian Mixture Regression and Classification*. PhD thesis, Rice University, Houston, Texas, 2004.

[91] J. Tang, A. Singh, N. Goehausen, and P. Abbeel. Parameterized maneuver learning for autonomous helicopter flight. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 1142–1148, May 2010.

[92] Y. Tang, R. Salakhutdinov, and G. Hinton. Deep mixtures of factor analysers. In *Proc. Intl Conf. on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.

[93] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999.

[94] E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5:1226–1235, 2002.

[95] K. Tokuda, T. Masuko, T. Yamada, T. Kobayashi, and S. Imai. An algorithm for speech parameter generation from continuous mixture HMMs with dynamic features. In *Proc. European Conference on Speech Communication and Technology (EUROSPEECH)*, pages 757–760, 1995.

[96] C. Towell, M. Howard, and S. Vijayakumar. Learning nullspace policies. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 241–248, 2010.

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

[98] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: TensorFaces. In *Computer Vision (ECCV)*, volume 2350 of *Lecture Notes in Computer Science*, pages 447–460. Springer Berlin Heidelberg, 2002.

[99] J. J. Verbeek, N. Vlassis, and B. Kroese. Efficient greedy learning of gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.

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

[101] Y. Wang and J. Zhu. DP-space: Bayesian nonparametric subspace clustering with small-variance asymptotics. In *Proc. Intl Conf. on Machine Learning (ICML)*, pages 1–9, Lille, France, 2015.

[102] A. D. Wilson and A. F. Bobick. Parametric hidden Markov models for gesture recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(9):884–900, 1999.

[103] D. M. Wolpert, J. Diedrichsen, and J. R. Flanagan. Principles of sensorimotor learning. *Nature Reviews*, 12:739–751, 2011.

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

[105] T. Yamazaki, N. Niwase, J. Yamagishi, and T. Kobayashi. Human walking motion synthesis based on multiple regression hidden semi-Markov model. In *Proc. Intl Conf. on Cyberworlds*, pages 445–452, 2005.

[106] H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech and Language*, 21(1):153–173, 2007.

[107] Q. Zhao, G. Zhou, T. Adali, L. Zhang, and A. Cichocki. Kernelization of tensor-based models for multiway data analysis: Processing of multidimensional structured data. *IEEE Signal Processing Magazine*, 30(4):137–148, 2013.