Learning Task Priorities from Demonstrations

João Silvério, Sylvain Calinon, Leonel Rozo, Darwin G. Caldwell

***Abstract*—Bimanual operations in humanoids offer the possibility to carry out more than one manipulation task at the same time, which in turn introduces the problem of task prioritization. We address this problem from a learning from demonstration perspective, by extending the Task-Parameterized Gaussian Mixture Model (TP-GMM) to Jacobian and null space structures. The proposed approach is tested on bimanual skills but can be applied in any scenario where the prioritization between potentially conflicting tasks needs to be learned. We evaluate the proposed framework in: two different tasks with humanoids requiring the learning of priorities and a loco-manipulation scenario, showing that the approach can be exploited to learn the prioritization of multiple tasks in parallel.**

***Index Terms*—learning from demonstration, task prioritization, bimanual manipulation.**

I. Introduction

THE human-robot transfer of bimanual skills is a growing topic of research in robotics. As the number of available dual-arm platforms and humanoid robots increases, existing learning and control algorithms must be reformulated to accommodate the constraints imposed by these morphologies and to take full advantage of the repertoire of tasks that such robots can perform [1]. Learning from Demonstration (LfD) [2] is a particularly promising direction to achieve a seamless transfer of bimanual abilities, but it has so far mostly addressed the learning of uni-manual and single-task skills. This article aims at extending the LfD paradigm to the learning of elaborated features that arise during bimanual manipulation in humanoids, particularly task prioritization.

As humans, we employ rich bimanual coordination behaviors on a daily basis (e.g., tying knots, moving heavy or bulky objects, sweeping). For this reason, most research on bimanual skill learning exploits operational space formulations (e.g. [3], [4], [5], [6], [7], [8], [9]), that focus on task space *constraints*, e.g. demonstrated coordination between end-effectors and object-related movements, that need to be reproduced precisely in order to successfully complete a task. However, constraints also arise in configuration space, for example as preferred body/arm postures or movements for which joint trajectories are more important than those of end-effectors. In such scenarios, operational space formulations alone are insufficient for correct task execution. Similarly, humanoids are often required to perform dexterous dual-arm skills that demand handling multiple potentially conflicting tasks in parallel. These conflicts can occur at various levels, such as when determining how to use the torso joints if both arms are needed (Fig. 1), or how to switch between poses while keeping balance. Endowing robots with the ability to learn how to handle priorities is an important research problem. It relates to the challenge of organizing movement primitives not only in series but also in parallel, which is often overlooked.

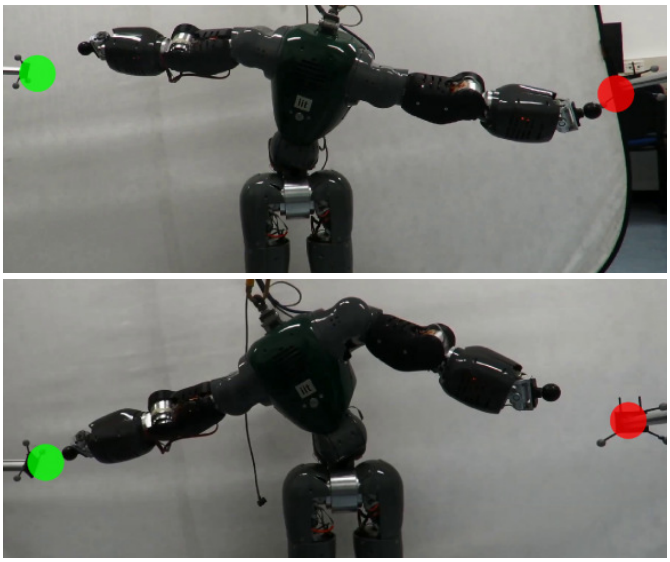


Fig. 1: The COMAN robot prioritizes the tracking of the left (resp. right) target, while doing its best to track the right (resp. left) one. This priority behavior was learned from demonstrations, using the approach proposed in Section V. **Top:** Reproduction with the model trained on left arm priority demonstrations. **Bottom:** Reproduction with the model trained on right arm priority demonstrations.

In this article, we offer a new perspective on task-parameterized movement models by including Jacobian matrices and null space structures in their formulation. Our approach takes Task-Parameterized Gaussian Mixture Models (TP-GMM) [8], [10] (reviewed in Section III) as an example, which was originally used to probabilistically encode Cartesian end-effector motions from the perspective of different coordinate systems. We propose here a more general formulation, exploiting the affine operations structure of TP-GMM to address the aforementioned LfD problems. More specifically, the contribution of this paper is a novel formulation of TP-GMM to:

1) **Simultaneously learn constraints in operational and configuration spaces (Section IV).** This article improves on previous work [11] in two directions: i) it formalizes the handling of operational and configuration space constraints in the context of TP-GMM and ii) it introduces

J. Silvério, L. Rozo and D. G. Caldwell are with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, 16163 Genova, Italy (e-mail: joao.silverio@iit.it; leonel.rozo@iit.it; darwin.caldwell@iit.it).

S. Calinon is with the Idiap Research Institute, CH-1920 Martigny, Switzerland, and with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, 16163 Genova, Italy (e-mail: sylvain.calinon@idiap.ch).

This work was supported by the MEMMO project (European Union's Horizon 2020 Programme, Grant 780684).

We would like to thank Luca Muratore and Phil Hudson for assisting with the COMAN experiments and Dr Martijn Zeestraten, Arturo Laurenzi and Giuseppe Rigano for the help with the Centauro simulator. We would also like to thank Dr Yanlong Huang for his feedback on previous versions of the paper.

unit quaternion-based projection operators, that permit the learning of orientation constraints.

2) **Learn task prioritization hierarchies (Section V).** Our formulation permits the identification of demonstrated priority behaviors, given an initial set of *candidate* task hierarchies. In addition, it allows the robot to reproduce the learned priorities in new situations. To the best of our knowledge, this is the first approach that permits learning full hierarchy structures from only few demonstrations.

Some of the aforementioned points were briefly introduced in [10]. Here we provide an extensive derivation, analysis and validation in three experimental scenarios, with real and simulated robotic platforms. First, we use the COMpliant huMANoid (COMAN) [12] robot to show that TP-GMM can be used to handle operational and configuration space constraints simultaneously. For this, we choose the skill of bimanually shaking a bottle (Section IV-D). We then use a bimanual reaching task to teach priorities from demonstrations (Section V-D). The task consists of tracking two conflicting targets on the left and right sides of COMAN with the corresponding arm. Finally, we consider a loco-manipulation scenario with the Centauro robot [13] in simulation (Section V-E). In this experiment, the prioritization of floating base position, end-effector positions and end-effector orientations needs to be learned, showing that the approach can be exploited in generic task prioritization applications, in particular with hierarchies of more than two tasks.

II. Related Work

A. Learning bimanual skills

The most popular approaches for learning bimanual manipulation from demonstrations are based on Dynamic Movement Primitives (DMPs) [14]. Examples range from the use of virtual springs between end-effectors [3] to the coupling of DMPs using artificial potential fields [4]. Lioutikov *et al.* [5] propose to combine sequences of DMPs that encode partial demonstrations of complete individual arm movements. Similarly to the spirit of DMPs, Likar *et al.* [6] introduce an approach based on Iterative Learning Control for force adaptation in bimanual tasks. In a more probabilistic fashion, Ureche and Billard [7] focus on the extraction of arm dominance and role from demonstrations, as well as on the correlations between task variables such as poses and forces. Our previous work [8], [9] follows a task-parameterized approach to learning bimanual skills, where relative and absolute end-effector movements are encoded with respect to a pre-defined set of coordinate systems, whose importance is learned probabilistically from demonstrations.

The foregoing collection of work addresses bimanual skill transfer from an operational space perspective. Consequently, rich features such as joint space movements and task prioritization, commonplace in highly redundant manipulators, cannot be adequately learned.

B. Simultaneous learning of operational and configuration space constraints

The problem of knowing which space—between configuration and operational spaces—is the most relevant for a given task, has frequently been treated as a hand-tuning of scalar weights assigned to sub-tasks in each space [15]. Exceptions include frameworks based on reinforcement learning (RL) [16], [17] and LfD [11], where the importance of each space is learned. Approaches like [16], [17] employ stochastic optimization, given a set of reward functions related to high level goals, to find optimal weights. In [11], the authors treat the problem as a weighted least squares problem where the weights associated with each space, encoded as full precision matrices, reflect the variability and correlations in the demonstrations. The two types of approach are similar in concept, with [11] and [16], [17] exploiting velocity and torque controllers, respectively. Despite the similarities, the RL-based approaches require that considerable prior knowledge is accounted for in the reward functions, making those methods less straightforward in many applications. In Section IV we propose an approach related to [11]. It improves on that work by considering an arbitrary number of tasks in operational space, as well as orientation constraints, which were previously overlooked.

Managing constraints from different spaces using full weight matrices, as we do in Section IV, can be seen as a form of task prioritization—that we refer to as a *fusion of tasks*. It consists of normally distributed control references that are fused as a product of Gaussians. This approach improves the traditional *soft weighting of tasks* (see Section II-C), where scalar weights are used to combine tasks with smooth transitions. The use of full matrices instead of scalar weights provides higher flexibility in the prioritization structure, while maintaining smoothness properties. Such approaches typically excel in tasks where the different constraints are activated sequentially, as we shall see in Section IV. Despite the advantages, they tend not to perform well when several constraints are activated at the same time, with similar weights, for which more elaborated prioritization structures are required. In the following section, we review different state-of-the-art techniques for learning task prioritization.

C. Learning task prioritization

Common approaches to control task prioritization can be categorized in two main directions, by either exploiting *strict hierarchy* structures, applied to multi-level hierarchies [18], [19], [20], or by employing a *soft weighting* of tasks [15], [21]. The two techniques have pros and cons. Setting an explicit null space structure guarantees strict priorities at the expense of constraining the tasks, which quickly limits the number of tasks that can simultaneously be handled with the number of degrees of freedom available for controlling the robot. This approach is also prone to discontinuities in the control problem when switching from one hierarchy structure to another. A soft weighting scheme can handle different levels of task importance and gradual changes from one task to another, but it does not provide strict guarantees on the fulfillment of each separated task.

In an alternative perspective, a collection of work addresses the problem using optimization [22], [23], [24] or by leveraging novel representations of prioritization [25], [26], [27].

In this article, we tackle the challenge from a robot learning perspective. Learning how to handle the priorities of multiple concurrent tasks is a challenging problem in robotics and, despite the recent advancements in this direction, several issues remain open.

1) Learning approaches based on strict hierarchies: Learning priorities based on strict task hierarchies typically assumes that low priority tasks are projected on the null space of high priority ones. In this context, Wrede *et al.* [28] propose a two-step approach to kinesthetically teach tasks to redundant manipulators. First, in a preliminary phase, the desired null space behavior is demonstrated to the gravity-compensated manipulator. The relation between end-effector positions and desired configurations is encoded in a neural network which is exploited during the demonstrations of the main task to control the robot null space. Saveriano *et al.* [29] propose an approach based on Task Transition Control (TTC) [30] to refine end-effector and null space policies. They take advantage of the smooth transitions between task priorities allowed by TTC to switch between task execution and teaching, yielding refinement of policies for both the main and the null space tasks in runtime. In [31], Towell *et al.* aim for extracting underlying null space policies from demonstrations, assuming a strict hierarchy of priorities, but the proposed approach does not allow for the extraction of demonstrated hierarchies. Lin *et al.* [32] propose an approach to learn the kinematic constraints present in movement observations, as explicitly represented by the null space projection matrix of kinematically constrained systems. A common assumption in both [31] and [32] is that one has access to the control variables during the demonstrations. This assumption is shared by our approach (Section V), in which we assume to know the control set-point associated with each sub-task.

Hak *et al.* [33] present an iterative algorithm for identifying a *stack of tasks*. From observed joint trajectories, and a pre-defined set of possible tasks that can be executed in parallel, the approach relies on the expected operational space behavior of each task to identify the active ones, and on the task-function formalism to gradually remove tasks until all have been identified. This approach shares similarities with ours in the assumption that demonstrated movements are generated from a strict hierarchy structure, and in the fact that it also analyses the operational space to disambiguate between possibly active tasks. The main difference is that our task space analysis is probabilistic, while Hak *et al.* rely on a curve fitting score. Moreover, their framework assumes prior knowledge about the possible tasks, while we assume prior knowledge about the possible task hierarchies, including the option to provide an exhaustive list of all possible ones. Finally, our probabilistic formulation permits the learning of input-dependent hierarchies, i.e. the robot can be taught priority behaviors that may vary during reproduction.

2) Approaches related to soft weighting of tasks: Concerning the soft weighting of tasks, control solutions are typically given by a combination of scalar-weighted sub-tasks, see [15] for an example with torque control and manually set weights. Dehio *et al.* [16] and Modugno *et al.* [17] propose to learn the weights of each sub-task using Covariance Matrix Adaptation Evolution Strategy (CMA-ES), a derivative-free stochastic optimization method. In [16], the weights are assumed to be constant throughout each execution, while in [17], they are parameterized by Radial Basis Functions spread along a time window, which allows the priorities of the different sub-tasks to change over time. Both approaches require the previous definition of a fitness function and a set of elementary tasks. Our approach also requires prior information about potential hierarchies, but, unlike [16], [17], it directly exploits the demonstrations to discover the employed prioritization, without requiring the defition of fitness functions.

	Learn hierarchies	From demonstrations	2+ tasks	Input dependent
Dehio *et al.* [16]	–	–	✓	–
Modugno *et al.* [17]	–	–	✓	✓
Wrede *et al.* [28]	–	✓	–	–
Hak *et al.* [33]	✓	✓	✓	–
Saveriano *et al.* [29]	–	✓	–	✓
Towell *et al.* [31]	–	✓	–	✓
Lin *et al.* [32]	✓	✓	–	–
Lober *et al.* [34], [35]	–	✓	–	✓
Paraschos *et al.* [36]	–	✓	✓	✓
Our approach	✓	✓	✓	✓

TABLE I: Contributions of our approach with respect to previous works on task priority learning.

Lober *et al.* [34] use stochastic optimization to refine DMPs of incompatible tasks in order to render them compatible. Their approach focuses on the re-organization of primitives in series, leaving out scenarios where they need to be executed in parallel with different levels of priority. In [35], the approach was improved by exploiting Gaussian kernels to compute variance-dependent weights that are used to determine the priority of different sub-tasks. This approach shares connections with ours in that variance is used as a measure of task importance. In that work, the variance depends on the distance to the kernel centers, which are pre-defined along the planned trajectory. In contrast, our method exploits the variability extracted from the demonstrations, allowing for learning new behaviors from the observed variations of a task.

Another approach along the lines of the above works is that of Paraschos *et al.* [36], who exploit the Probabilistic Movement Primitives (ProMP) framework to learn the accuracy of different tasks. This information, which reflects the variability of demonstrations given in Cartesian and joint spaces, is then used as a prioritization criteria to organize the different tasks. In this sense, that work shares stronger connections with [11] than with our approach in Section V as it does not generate strict hierarchies to reproduce demonstrated movements. By identifying the hierarchies employed in the demonstrations, our approach ensures a more strict fulfillment of each individual task.

Table I summarizes the main contributions of our approach for learning task prioritization hierarchies, described in Section V, with respect to the state-of-the-art. The column *Learn hierarchies* indicates whether or not the approach outputs strict hierarchies, as opposed to weights (scalar or matrix), while the column *From demonstrations* distinguishes between LfD and RL approaches.

III. Task-Parameterized Gaussian Mixture Models

In previous work [8], [10] we introduced a probabilistic approach to the learning of task-parameterized movement primitives, with an example of implementation as Task-Parameterized Gaussian Mixture Model (TP-GMM). TP-GMM was used to encode demonstrated end-effector motions in multiple coordinate systems simultaneously, whose importance changed during the task depending on the variability observed in the demonstrations. This information was exploited to adapt skills to new situations, in accordance to the demonstrated local features. In this section we review the TP-GMM formulation.

A. Overview and nomenclature

Throughout this article we will exploit extensively the linear properties of Gaussian distributions, which are central to TP-GMM. For this reason, in this subsection we review the concept of so-called *task parameters* and give an overview of the technique.

Definition 1. *Task parameters are sets of linear operators $\boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}$ that map Gaussian distributions from P subspaces, indexed by $j = 1, \dots, P$, onto a common space. Such sets are called "task parameters" since they are part of the parameterization of a task, i.e., they influence how the robot accomplishes the given task.*

In TP-GMM, P subspaces encode local features of a demonstrated skill. Once projected onto a common space, through *task parameters*, the local models are combined to provide a solution fulfilling the most important features during task execution (Section III-C). In previous work, task parameters have been used to represent poses of objects in a robot workspace, mapping local models of demonstrations (from the perspective of P different objects) onto a global coordinate system, typically the robot base frame. In this case, $\boldsymbol{A}^{(j)}$ is a rotation matrix [8], [10], [37], [38] or a quaternion matrix [9], representing an object orientation, and $\boldsymbol{b}^{(j)}$ is a translation vector, representing the origin of an object coordinate system with respect to the base frame of the robot. It was common in previous work to refer to task parameters as *candidate frames* or *candidate coordinate systems*. This is because, for a given task, each set of task parameters $\boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}$ may or may not influence the task execution, depending on the variability of the teacher's demonstrations in the corresponding coordinate system. Hence, each set $\boldsymbol{A}^{(j)}, \boldsymbol{b}^{(j)}$ is considered to be a *candidate* for affecting the task outcome.

In this work, we exploit the linear structure inherent to the task parameter formulation of TP-GMM to address the problems of: 1) simultaneously learning operational and configuration space constraints; and 2) learning priority hierarchies from demonstrations. We do this by taking a different perspective from past work where task parameters were candidate coordinate systems. We propose a formulation of task parameters that correspond to *candidate projection operators* (Section IV) and *candidate task hierarchies* (Section V).

For clarity, Table II summarizes the most important variables and symbols that are exploited throughout the article.

	Variable / symbol	Description
TP-GMM	P	Number of task parameters
	K	Number of Gaussian components
	$\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$	Task parameters $j \in [1, P]$
	$\boldsymbol{\mu}_i^{(j)}$, $\boldsymbol{\Sigma}_i^{(j)}$	Mean and covariance of Gaussian $i \in [1, K]$ for task parameter j
	π_i	Mixing coefficient of Gaussian i
	M	Number of demonstrations
	T_m	Number of datapoints in demonstration $m \in [1, M]$
	N	Total number of datapoints in a demonstration dataset
	D	Dimension of datapoints
	$\boldsymbol{\xi}_t \in \mathbb{R}^D$	Datapoint at time t
	$\boldsymbol{\xi} \in \mathbb{R}^{D \times N}$	Demonstration dataset
Robot	$\boldsymbol{x}_L, \boldsymbol{x}_R \in \mathbb{R}^3$	Cartesian position of left and right end-effectors
	$\boldsymbol{\epsilon}_L, \boldsymbol{\epsilon}_R \in \mathcal{S}^3$	Orientation of left and right end-effectors as unit quaternions
	$\boldsymbol{R}_L, \boldsymbol{R}_R \in \mathrm{SO}(3)$	Orientation of left and right end-effectors as rotation matrices
	N_q	Number of robot joints
	$\boldsymbol{q}_L, \boldsymbol{q}_R \in \mathbb{R}^{N_q}$	Joint angles of left and right arms
	$\boldsymbol{J}_L, \boldsymbol{J}_R \in \mathbb{R}^{6 \times N_q}$	Jacobian matrices of left and right arms
	$>$	Priority operator (the task on the operator left has *priority over* the one on the right)

TABLE II: Summary of the notation.

B. Model estimation

Each demonstration $m \in \{1, \dots, M\}$ contains T_m datapoints of dimension D forming a dataset of N datapoints $\{\boldsymbol{\xi}_t\}_{t=1}^N$ with $N = \sum_{m=1}^M T_m$ and $\boldsymbol{\xi}_t \in \mathbb{R}^D$. P task parameters, that map between subspaces $j = 1, \dots, P$ and a common space, are defined at every time step t by $\{\boldsymbol{A}_t^{(j)}, \boldsymbol{b}_t^{(j)}\}_{j=1}^P$. The demonstrations $\boldsymbol{\xi} \in \mathbb{R}^{D \times N}$ are observed from each subspace, forming P local datasets $\boldsymbol{X}^{(j)} \in \mathbb{R}^{D \times N}$.*

The *model parameters* of a TP-GMM with K components are defined by $\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^P\}_{i=1}^K$, where π_i are the mixing coefficients and $\boldsymbol{\mu}_i^{(j)}$, $\boldsymbol{\Sigma}_i^{(j)}$ denote the center and covariance matrix of the i-th Gaussian in subspace j. Learning of the model parameters is achieved by log-likelihood maximization using an *expectation-maximization* (EM) algorithm, see [10] for details.

C. Gaussian Mixture Regression

The learned model is used to reproduce movements in new situations. Each subspace j encodes local features of the demonstrated movement. In new situations, i.e., for new values of the task parameters $\boldsymbol{A}_t^{(j)}, \boldsymbol{b}_t^{(j)}$, one needs to find a trade-off between each subspace solution. TP-GMM solves this problem by fusing the local models through projection in a common space, using the product of Gaussians. In this way, a new GMM with parameters $\{\pi_i, \hat{\boldsymbol{\mu}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\}_{i=1}^K$ is automatically

*As an example, in previous work [8], [10], [38], $\boldsymbol{\xi}_t$ corresponded to end-effector positions, and local datasets could be computed with $\boldsymbol{X}_t^{(j)} = \boldsymbol{A}_t^{(j)^{-1}} \left(\boldsymbol{\xi}_t - \boldsymbol{b}_t^{(j)}\right)$, with task parameters $\{\boldsymbol{A}_t^{(j)}, \boldsymbol{b}_t^{(j)}\}_{j=1}^P$ representing coordinate systems parameterized by the orientations and positions of P objects.

generated as

$$\mathcal{N}\Big(\hat{\boldsymbol{\mu}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\Big) \propto \prod_{j=1}^{P} \mathcal{N}\Big(\hat{\boldsymbol{\mu}}_{t,i}^{(j)},\ \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)}\Big), \text{ with}$$
$$\hat{\boldsymbol{\mu}}_{t,i}^{(j)} = \boldsymbol{A}_t^{(j)}\boldsymbol{\mu}_i^{(j)} + \boldsymbol{b}_t^{(j)}\ , \quad \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)} = \boldsymbol{A}_t^{(j)}\boldsymbol{\Sigma}_i^{(j)}\boldsymbol{A}_t^{(j)\mathsf{T}}, \quad (1)$$

where the Gaussian product is analytically given by

$$\hat{\boldsymbol{\Sigma}}_{t,i} = \Big(\sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)-1}\Big)^{-1}, \quad \hat{\boldsymbol{\mu}}_{t,i} = \hat{\boldsymbol{\Sigma}}_{t,i}\sum_{j=1}^{P} \hat{\boldsymbol{\Sigma}}_{t,i}^{(j)-1}\hat{\boldsymbol{\mu}}_{t,i}^{(j)}. \quad (2)$$

Equation (1) maps local models onto a common space, where information from the different subspaces is fused according to Eq. (2). Note that the task parameters $\boldsymbol{A}_t^{(j)}, \boldsymbol{b}_t^{(j)}$ may vary during reproduction and take values different from those observed during demonstrations. In previous work [8], [9], [37], [38], this property was exploited to adapt demonstrated skills to new situations (typically, new position and orientation of manipulated objects).

The obtained GMM is used to generate a reference trajectory distribution for the robot through Gaussian Mixture Regression (GMR). In this case, the datapoint $\boldsymbol{\xi}_t$ is decomposed into two subvectors $\boldsymbol{\xi}_t^{\mathcal{I}}$ and $\boldsymbol{\xi}_t^{\mathcal{O}}$, spanning the input and output dimensions of the regression problem, thus the GMM obtained from (1) and (2) encodes the joint probability distribution $\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{I}}, \boldsymbol{\xi}_t^{\mathcal{O}}) \sim \sum_{i=1}^{K} \pi_i\, \mathcal{N}\big(\hat{\boldsymbol{\mu}}_{t,i}, \hat{\boldsymbol{\Sigma}}_{t,i}\big)$. For a time-driven movement, $\boldsymbol{\xi}_t^{\mathcal{I}}$ corresponds to the current time step, while $\boldsymbol{\xi}_t^{\mathcal{O}}$ can be the end-effector pose or the joint angles of the robot. The task parameters $\boldsymbol{A}_t^{(j)}$ and $\boldsymbol{b}_t^{(j)}$ are also decomposed so that the input is not modulated by the task parameterization. Compared to an initial TP-GMM encoding $\boldsymbol{\xi}_t^{\mathcal{O}}$ with task parameters $\boldsymbol{A}_t^{\mathcal{O}(j)}$ and $\boldsymbol{b}_t^{\mathcal{O}(j)}$, the combination of TP-GMM and GMR instead encodes $\boldsymbol{\xi}_t = \big[\boldsymbol{\xi}_t^{\mathcal{I}\mathsf{T}}\ \ \boldsymbol{\xi}_t^{\mathcal{O}\mathsf{T}}\big]^{\mathsf{T}}$ with[†]

$$\boldsymbol{A}_t^{(j)} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_t^{\mathcal{O}(j)} \end{bmatrix}, \quad \boldsymbol{b}_t^{(j)} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{b}_t^{\mathcal{O}(j)} \end{bmatrix}. \quad (3)$$

Thus, GMR generates a new distribution $\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \mathcal{N}\big(\boldsymbol{\xi}_t^{\mathcal{O}}|\hat{\boldsymbol{\mu}}_t^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}}\big)$ that is used to control the robot.

IV. Learning operational and configuration space constraints simultaneously

In this section, we propose to exploit the structure of TP-GMM introduced in Section III to simultaneously consider constraints in operational and configuration spaces. The aim is to circumvent the need for previously selecting the space in which one should encode a given task, and instead let the model automatically discover the proper space from a small set of demonstrations. The approach consists of encoding the demonstrated movement in both spaces and, through statistics, extracting the space with the least variability (i.e., with the highest consistency) at each reproduction step. We do this by considering Jacobian-based task parameters (formulated in Sections IV-A and IV-B) that project operational space constraints onto configuration space (Section IV-C), where Gaussian products (1) are computed.

[†]in the case of a scalar time input, the identity matrix $\boldsymbol{I}$ collapses to 1.

Conceptually, this approach shares connections with the one introduced in [11]. However, here we:

1) consider the learning of operational space constraints with respect to an arbitrary number of objects, by framing the approach in the context of TP-GMM, and
2) develop a formulation to consider end-effector orientation constraints.

Additionally, the analysis focuses on humanoid robots and bimanual manipulation.

A. Jacobian-based task parameters for position constraints

Handling constraints in configuration and operational spaces is achieved by exploiting the task parameter linear structure in TP-GMM. Formally, consider a manipulator with N_q joints, whose positions and velocities are denoted by $\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^{N_q}$. Its differential kinematics are given by $\begin{bmatrix}\dot{\boldsymbol{x}}^{\mathsf{T}}\ \boldsymbol{\omega}^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}} = \boldsymbol{J}\dot{\boldsymbol{q}}$, where $\dot{\boldsymbol{x}}, \boldsymbol{\omega} \in \mathbb{R}^3$ are the operational space linear and angular velocities. The Jacobian matrix $\boldsymbol{J} = \begin{bmatrix}\boldsymbol{J}_p^{\mathsf{T}}\ \boldsymbol{J}_o^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{6\times N_q}$ accounts for the contribution of joint velocities to operational space velocities, with matrices $\boldsymbol{J}_p, \boldsymbol{J}_o \in \mathbb{R}^{3\times N_q}$ responsible for the linear and angular parts, respectively. We here assume redundant manipulators, i.e. $N_q > 6$.

The inverse differential kinematics for the position part of the operational space is given by $\dot{\boldsymbol{q}} = \boldsymbol{J}_p^{\dagger}\dot{\boldsymbol{x}}$, with $\boldsymbol{J}_p^{\dagger} = \boldsymbol{J}_p^{\mathsf{T}}\Big(\boldsymbol{J}_p\boldsymbol{J}_p^{\mathsf{T}}\Big)^{-1}$ the right pseudo-inverse of the Jacobian $\boldsymbol{J}_p$. It yields the minimum-norm $\dot{\boldsymbol{q}}$ that ensures $\dot{\boldsymbol{x}}$ in operational space [39]. Numerical integration of this equation permits the computation of joint references for a desired end-effector position $\boldsymbol{x}_t$ as (dropping the Jacobian subscript p)

$$\begin{aligned} &\hat{\boldsymbol{q}}_t - \boldsymbol{q}_{t-1} = \boldsymbol{J}_{t-1}^{\dagger}\left(\boldsymbol{x}_t - \boldsymbol{x}_{t-1}\right) \\ &\iff \hat{\boldsymbol{q}}_t = \boldsymbol{J}_{t-1}^{\dagger}\boldsymbol{x}_t + \boldsymbol{q}_{t-1} - \boldsymbol{J}_{t-1}^{\dagger}\boldsymbol{x}_{t-1}, \end{aligned} \quad (4)$$

where $\hat{\boldsymbol{q}}_t$ denotes the desired joint angles at t. The structure of (4), being affine in $\boldsymbol{x}_t$, allows us to connect inverse kinematics with TP-GMM. Let us assume, for the sake of the argument, that end-effector positions are modeled according to $\boldsymbol{x}_t \sim \sum_{i=1}^{K} \pi_i \mathcal{N}\left(\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\right)$. The index j denotes one arbitrary subspace, as discussed in Section III, where robot end-effector positions are locally modeled by a GMM. It follows that the linear transformation properties of Gaussian distributions (1) can be applied to (4) to project the local GMM onto the configuration space, resulting in

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \underbrace{\boldsymbol{J}_{t-1}^{\dagger}}_{\boldsymbol{A}_t^{(j)}}\boldsymbol{\mu}_i^{(j)} + \underbrace{\boldsymbol{q}_{t-1} - \boldsymbol{J}_{t-1}^{\dagger}\boldsymbol{x}_{t-1}}_{\boldsymbol{b}_t^{(j)}}, \quad \forall i = 1, \dots, K, \quad (5)$$

for the center of the Gaussian i, and

$$\hat{\boldsymbol{\Sigma}}_{t,i}^{(j)} = \underbrace{\boldsymbol{J}_{t-1}^{\dagger}}_{\boldsymbol{A}_t^{(j)}}\boldsymbol{\Sigma}_i^{(j)}\underbrace{(\boldsymbol{J}_{t-1}^{\dagger})^{\mathsf{T}}}_{\boldsymbol{A}_t^{(j)\mathsf{T}}}, \quad \forall i = 1, \dots, K, \quad (6)$$

for the corresponding covariance matrix. Equations (5) and (6) show that Jacobian-based task parameters $\boldsymbol{A}_t^{(j)}, \boldsymbol{b}_t^{(j)}$ act as *projection operators* that map Gaussian distributions from operational to configuration space, creating distributions of joint angles.

This result is the cornerstone of more complex types of operators. For instance, if we consider end-effector positions encoded with respect to an object parameterized by a translation vector $\boldsymbol{p}_t^{(j)}$ and a rotation matrix $\boldsymbol{R}_t^{(j)}$, (5) becomes

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \underbrace{\boldsymbol{J}_{t-1}^{\dagger}\boldsymbol{R}_t^{(j)}}_{\boldsymbol{A}_t^{(j)}}\boldsymbol{\mu}_i^{(j)} + \underbrace{\boldsymbol{J}_{t-1}^{\dagger}\left(\boldsymbol{p}_t^{(j)} - \boldsymbol{x}_{t-1}\right) + \boldsymbol{q}_{t-1}}_{\boldsymbol{b}_t^{(j)}}, \quad (7)$$

which can be derived in a straightforward manner from (5) by assuming rotated and translated Gaussians with center $\boldsymbol{R}_t^{(j)}\boldsymbol{\mu}_i^{(j)} + \boldsymbol{p}_t^{(j)}$ and covariance $\boldsymbol{R}_t^{(j)}\boldsymbol{\Sigma}_i^{(j)}\boldsymbol{R}_t^{(j)^{\mathsf{T}}}$. Similarly, the expression for the covariance matrix $\hat{\boldsymbol{\Sigma}}_{t,i}^{(j)}$ can be easily obtained based on (6), employing $\boldsymbol{A}_t^{(j)}$ as defined in (7).

On the basis of this construction of task parameters, the local datasets $\boldsymbol{X}^{(j)} = [\boldsymbol{X}_1^{(j)}, \dots, \boldsymbol{X}_N^{(j)}]$ used to train the TP-GMM model can be constructed in a similar fashion as in Section III. If a subspace j models the absolute end-effector position, i.e. with respect to the base frame of the robot, we have $\boldsymbol{X}_t^{(j)} = \boldsymbol{x}_t$. On the other hand, when j is associated with a coordinate system with pose parameters $\boldsymbol{p}_t^{(j)}, \boldsymbol{R}_t^{(j)}$ as in (7), we have $\boldsymbol{X}_t^{(j)} = \boldsymbol{R}_t^{(j)^{\mathsf{T}}}\left(\boldsymbol{x}_t - \boldsymbol{p}_t^{(j)}\right)$.

In summary, in this novel formulation of task parameters, $\boldsymbol{A}_t^{(j)} \in \mathbb{R}^{N_q \times 3}$ and $\boldsymbol{b}_t^{(j)} \in \mathbb{R}^{N_q}$ map from Cartesian position to joint angles, solving the inverse kinematics with a reference given by the mean $\boldsymbol{\mu}_i^{(j)}$. The TP-GMM representation is therefore extended to Jacobian-based, time-varying, task parameters $\{\boldsymbol{b}_t^{(j)}, \boldsymbol{A}_t^{(j)}\}_{j=1}^{P}$ with non-square $\boldsymbol{A}_t^{(j)}$ matrices. With this representation, operational space constraints are projected onto configuration space, where Gaussian products can be computed as in (2), extending the original TP-GMM formulation to the consideration of configuration space constraints.

B. Task parameters for orientation

Orientation constraints represent an important component of operational space in many tasks and in bimanual manipulation the orientation between end-effectors is of utmost importance for correct task execution [9]. Here, we take advantage of the algebraic properties of unit quaternions to derive linear operators for projecting orientation constraints onto configuration space.

Let us consider the orientation part of the end-effector pose represented by a unit quaternion $\boldsymbol{\epsilon}$ (Appendix A reviews this representation[‡]) and the inverse differential kinematics for angular velocities, $\dot{\boldsymbol{q}} = \boldsymbol{J}_o^{\dagger}\boldsymbol{\omega}$. From [40] we have that

$$\boldsymbol{\omega}_t \approx \frac{\mathrm{vec}(\boldsymbol{\epsilon}_t * \bar{\boldsymbol{\epsilon}}_{t-1})}{\Delta t} \quad (8)$$

gives the angular velocity that rotates the unit quaternion $\boldsymbol{\epsilon}_{t-1}$ into $\boldsymbol{\epsilon}_t$, during Δt. The operation $\mathrm{vec}(\boldsymbol{\epsilon}_t * \bar{\boldsymbol{\epsilon}}_{t-1})$ can be replaced by the matrix-vector product $\bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{t-1})\,\boldsymbol{\epsilon}_t$ (see Appendix B), allowing us to write the inverse kinematics equation as (dropping the subscript o in the Jacobian matrix)

$$\hat{\dot{\boldsymbol{q}}}_t = \boldsymbol{J}_{t-1}^{\dagger}\bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{t-1})\,\boldsymbol{\epsilon}_t\frac{1}{\Delta t}\,, \quad (9)$$

$$\iff \hat{\boldsymbol{q}}_t = \boldsymbol{J}_{t-1}^{\dagger}\bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{t-1})\,\boldsymbol{\epsilon}_t + \boldsymbol{q}_{t-1}, \quad (10)$$

[‡]The appendices can be found as supplementary material at `http://joaosilverio.weebly.com/tro`

Algorithm 1 Simultaneously learning constraints in operational and configuration spaces

Initialization

1: Select candidate projection operators from Table III based on the task at hand
- Canonical operator $\boldsymbol{A}_t^{(j)} = \boldsymbol{I}$, $\boldsymbol{b}_t^{(j)} = \boldsymbol{0}$, for encoding configuration space constraints
- Operational space operators, for absolute or relative position/orientation constraints in operational space

2: Collect demonstrations and compute the local datasets $\boldsymbol{X}^{(j)}$ according to the chosen operators

Model training

1: Apply EM [10] to obtain $\left\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\right\}_{i=1}^{K}$

Movement synthesis

1: **for** $t = 1, \dots, N$ **do**
2: **for** $j = 1, \dots, P$ **do**
3: Update $\{\boldsymbol{A}_t^{(j)}, \boldsymbol{b}_t^{(j)}\}$ according to Table III
4: **end for**
5: **for** $i = 1, \dots, K$ **do**
6: Compute $\hat{\boldsymbol{\mu}}_{t,i}$ and $\hat{\boldsymbol{\Sigma}}_{t,i}$ from (1) and (2)
7: **end for**
8: Apply GMR at $\boldsymbol{\xi}_t^{\mathcal{I}}$: $\mathcal{P}(\boldsymbol{\xi}_t^{\mathcal{O}}|\boldsymbol{\xi}_t^{\mathcal{I}}) = \mathcal{N}\left(\boldsymbol{\xi}_t^{\mathcal{O}}|\hat{\boldsymbol{\mu}}_t^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_t^{\mathcal{O}}\right)$
9: Use $\hat{\boldsymbol{\mu}}_t^{\mathcal{O}}$ as joint references for the robot controller
10: **end for**

which has a similar structure to (4), being linear for the quaternion $\boldsymbol{\epsilon}_t$. In a similar way as in Section IV-A, if $\boldsymbol{\mu}_i^{(j)}$ is the center of a Gaussian i, encoding the absolute orientation of the end-effector in a subspace j, we take advantage of the structure in (10) to devise new task parameters $\boldsymbol{A}_t^{(j)}$, $\boldsymbol{b}_t^{(j)}$ that map a GMM from quaternion space to configuration space, namely

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \underbrace{\boldsymbol{J}_{t-1}^{\dagger}\bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{t-1})}_{\boldsymbol{A}_t^{(j)}}\,\boldsymbol{\mu}_i^{(j)} + \underbrace{\boldsymbol{q}_{t-1}}_{\boldsymbol{b}_t^{(j)}}, \quad (11)$$

for the center of Gaussian i in subspace j (the covariance can be derived by following the same rules that we explained in Section IV-A).

For a desired end-effector orientation encoded in a coordinate system whose orientation is given by $\boldsymbol{\epsilon}_t^{(j)}$, (11) becomes

$$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \underbrace{\boldsymbol{J}_{t-1}^{\dagger}\bar{\boldsymbol{H}}^{*}\left(\bar{\boldsymbol{\epsilon}}_{t-1}\right)\overset{+}{\boldsymbol{H}}\left(\boldsymbol{\epsilon}_t^{(j)}\right)}_{\boldsymbol{A}_t^{(j)}}\boldsymbol{\mu}_i^{(j)} + \underbrace{\boldsymbol{q}_{t-1}}_{\boldsymbol{b}_t^{(j)}}, \quad (12)$$

where $\overset{+}{\boldsymbol{H}}$ is a quaternion matrix (see Appendix A).

For this formulation, if a subspace j models the absolute end-effector orientation, i.e. with respect to the base frame of the robot, we have $\boldsymbol{X}_t^{(j)} = \boldsymbol{\epsilon}_t$. When j is associated with a coordinate system with quaternion $\boldsymbol{\epsilon}_t^{(j)}$, then $\boldsymbol{X}_t^{(j)} = \overset{+}{\boldsymbol{H}}\left(\bar{\boldsymbol{\epsilon}}_t^{(j)}\right)\boldsymbol{\epsilon}_t$.

C. Task parameters for configuration space constraints

The previous two subsections provided task parameters that project operational space constraints onto configuration space. However, in order to consider both operational and configuration space constraints, one requires a local model of the configuration space demonstrations as well. Encoding configuration space movements in a TP-GMM is done using simple task parameters $\boldsymbol{A}_t^{(j)} = \boldsymbol{I}$, $\boldsymbol{b}_t^{(j)} = \boldsymbol{0}$, corresponding

TABLE III: Summary of task parameters as candidate projection operators

Configuration space constraints:	$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{I}$	$\boldsymbol{\mu}_i^{(j)} + \mathbf{0}$
Absolute position constraints:	$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{J}_{t-1}^{\dagger}$	$\boldsymbol{\mu}_i^{(j)} - \boldsymbol{J}_{t-1}^{\dagger}\boldsymbol{x}_{t-1} + \boldsymbol{q}_{t-1}$
Relative position constraints:	$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{J}_{t-1}^{\dagger}\boldsymbol{R}_t^{(j)}$	$\boldsymbol{\mu}_i^{(j)} + \boldsymbol{J}_{t-1}^{\dagger}(\boldsymbol{p}_t^{(j)} - \boldsymbol{x}_{t-1}) + \boldsymbol{q}_{t-1}$
Absolute orientation constraints:	$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{J}_{t-1}^{\dagger}\bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{t-1})$	$\boldsymbol{\mu}_i^{(j)} + \boldsymbol{q}_{t-1}$
Relative orientation constraints:	$\hat{\boldsymbol{q}}_{t,i}^{(j)} = \underbrace{\boldsymbol{J}_{t-1}^{\dagger}\bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{t-1})\overset{+}{\bar{\boldsymbol{H}}}(\boldsymbol{\epsilon}_t^{(j)})}_{\boldsymbol{A}_t^{(j)}}$	$\boldsymbol{\mu}_i^{(j)} + \underbrace{\boldsymbol{q}_{t-1}}_{\boldsymbol{b}_t^{(j)}}$

to a canonical projection operator. In this case, the subspace is the configuration space itself, i.e. , $\hat{\boldsymbol{q}}_{t,i}^{(j)} = \boldsymbol{\mu}_i^{(j)}$. The local datasets are thus computed from $\boldsymbol{X}_t^{(j)} = \boldsymbol{q}_t$, where $\boldsymbol{q}_t \in \mathbb{R}^{N_q}$ is the vector of robot joint angles at time step t.

Table III gives a summary of the operators derived in this section. The overall procedure for learning and reproducing a skill is summarized in Algorithm 1.

D. Experiment: bimanual shaking skill

In order to test the formulation introduced in this section, we selected the skill of shaking a bottle using COMAN. The skill contains an operational space component (reaching, grasping a bottle and bringing it closer to the torso) and a configuration space component (shaking with rhythmic shoulder movements, see Fig. 2). Parts of this experiment are also reported in [41].

The upper-body of the COMAN robot comprises 17 DOFs: 3 DOFs for the waist and 7 for each arm, with the kinematic chains of both arms sharing the 3 waist joints. We define the differential kinematics of the left and right end-effectors as $\begin{bmatrix}\dot{\boldsymbol{x}}_L^{\mathsf{T}} & \boldsymbol{\omega}_L^{\mathsf{T}} & \dot{\boldsymbol{x}}_R^{\mathsf{T}} & \boldsymbol{\omega}_R^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}} = \boldsymbol{J}_{\text{up}}\dot{\boldsymbol{q}}$, where $\boldsymbol{J}_{\text{up}}$ is the upper-body Jacobian, $\dot{\boldsymbol{x}}_L, \boldsymbol{\omega}_L$, $\dot{\boldsymbol{x}}_R, \boldsymbol{\omega}_R$ are the left and right end-effector velocities and $\dot{\boldsymbol{q}} = \begin{bmatrix}\dot{\boldsymbol{q}}_W^{\mathsf{T}} & \dot{\boldsymbol{q}}_L^{\mathsf{T}} & \dot{\boldsymbol{q}}_R^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}$ represents the concatenation of waist, left and right arm joint velocities. The Jacobian matrix is given by [42]

$$\boldsymbol{J}_{\text{up}} = \begin{bmatrix}\boldsymbol{J}_{W|L} & \boldsymbol{J}_L & \mathbf{0} \\ \boldsymbol{J}_{W|R} & \mathbf{0} & \boldsymbol{J}_R\end{bmatrix}, \tag{13}$$

where $\boldsymbol{J}_{W|L}$, $\boldsymbol{J}_{W|R}$ denote the Jacobians that account for the effect of the waist joints on left and right end-effector velocities. $\boldsymbol{J}_L$ and $\boldsymbol{J}_R$ correspond to the Jacobians of the left and right end-effectors from the waist link. The inverse kinematics solution is given in this case by $\hat{\dot{\boldsymbol{q}}} = \begin{bmatrix}\hat{\dot{\boldsymbol{q}}}_W^{\mathsf{T}} & \hat{\dot{\boldsymbol{q}}}_L^{\mathsf{T}} & \hat{\dot{\boldsymbol{q}}}_R^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}} = \boldsymbol{J}_{\text{up}}^{\dagger}\begin{bmatrix}\dot{\boldsymbol{x}}_L^{\mathsf{T}} & \boldsymbol{\omega}_L^{\mathsf{T}} & \dot{\boldsymbol{x}}_R^{\mathsf{T}} & \boldsymbol{\omega}_R^{\mathsf{T}}\end{bmatrix}^{\mathsf{T}}$, where $\boldsymbol{J}_{\text{up}}^{\dagger}$ is the right pseudo-inverse of $\boldsymbol{J}_{\text{up}}$.

The experiment was conducted in the Gazebo simulator, and the skill was reproduced in the real robot (Figure 3). The demonstrations were generated by solving inverse kinematics in Gazebo, for the operational space part of the movement, and using sinusoidal references to control the shoulder joints for the shaking part[‡]. Here, we make the assumption that the demonstrated grasp is always successful and the object will move together with the end-effectors after it is grasped. Hence, the pose of the bottle that is considered in this experiment is the one at the beginning of each demonstration. We also assume that the grasping points on the bottle are the same in all demonstrations. We collected 10 demonstrations of the skill, each of them with different initial bottle poses and a duration of approximately 13 seconds. In each demonstration, we recorded both the joint angles and the end-effector poses with respect to the initial bottle frame. Temporal alignment of the demonstrations was achieved using Dynamic Time Warping [43]. We used a TP-GMM with $K = 10$ components, with $P = 2$ projection operators (K and P chosen empirically). The first operator is a concatenation of (7), required for considering end-effector positions, and (12), for orientations, parameterized with the bottle pose $\{\boldsymbol{p}_t^{(1)}, \boldsymbol{R}_t^{(1)}, \boldsymbol{\epsilon}_t^{(1)}\}$,

$$\boldsymbol{A}_t^{(1)} = \boldsymbol{J}_{\text{up}}^{\dagger}\begin{bmatrix}\boldsymbol{R}_t^{(1)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{L,t-1})\overset{+}{\bar{\boldsymbol{H}}}(\boldsymbol{\epsilon}_t^{(1)}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{R}_t^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \bar{\boldsymbol{H}}^{*}(\bar{\boldsymbol{\epsilon}}_{R,t-1})\overset{+}{\bar{\boldsymbol{H}}}(\boldsymbol{\epsilon}_t^{(1)})\end{bmatrix}, \tag{14}$$

$$\boldsymbol{b}_t^{(1)} = \boldsymbol{J}_{\text{up}}^{\dagger}\begin{bmatrix}\boldsymbol{p}_t^{(1)} - \boldsymbol{x}_{L,t-1} \\ \mathbf{0} \\ \boldsymbol{p}_t^{(1)} - \boldsymbol{x}_{R,t-1} \\ \mathbf{0}\end{bmatrix} + \boldsymbol{q}_{t-1}. \tag{15}$$

For these task parameters, training datapoints are concatenations of both end-effector poses from the perspective of the bottle frame, i.e.

$$\boldsymbol{X}_t^{(1)} = \begin{bmatrix}\boldsymbol{R}_t^{(1)^{\mathsf{T}}}\left(\boldsymbol{x}_{L,t}^{\mathsf{T}} - \boldsymbol{p}_t^{(1)}\right) \\ \overset{+}{\bar{\boldsymbol{H}}}\left(\bar{\boldsymbol{\epsilon}}_t^{(1)}\right)\boldsymbol{\epsilon}_{L,t}^{\mathsf{T}} \\ \boldsymbol{R}_t^{(1)^{\mathsf{T}}}\left(\boldsymbol{x}_{R,t}^{\mathsf{T}} - \boldsymbol{p}_t^{(1)}\right) \\ \overset{+}{\bar{\boldsymbol{H}}}\left(\bar{\boldsymbol{\epsilon}}_t^{(1)}\right)\boldsymbol{\epsilon}_{R,t}^{\mathsf{T}}\end{bmatrix}. \tag{16}$$

The second projection operator is canonical, $\boldsymbol{A}_t^{(2)} = \boldsymbol{I}$, $\boldsymbol{b}_t^{(2)} = \mathbf{0}$. In the above, subscripts L and R denote left and right end-effectors.

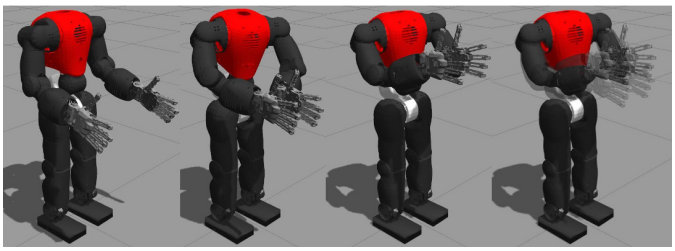


Fig. 2: The COMAN robot performs the bimanual shaking task in simulation. **First:** The robot is in a neutral starting pose. **Second:** Reaching for the bottle and grasping it. **Third:** Bringing the bottle close to the torso. **Fourth:** Shaking movement executed through rhythmic oscillations of both shoulder joints.

[‡]Alternatively, kinesthetic teaching or optical tracking of movements from humans could be used.

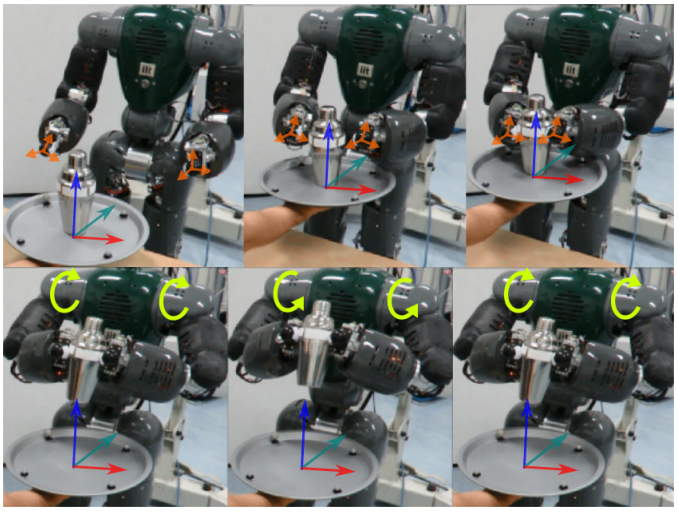


Fig. 3: Reproduction of the shaking task in the real COMAN robot. **Top:** Snapshots of the reaching part of the movement, defined by constraints in operational space. The robot gradually reaches for the object, which is tracked using an optical system. **Bottom:** Shaking part of the movement, defined by constraints in configuration space. The robot performs the shaking through rhythmic motions of the shoulders, moving the shaker up and down repeatedly.

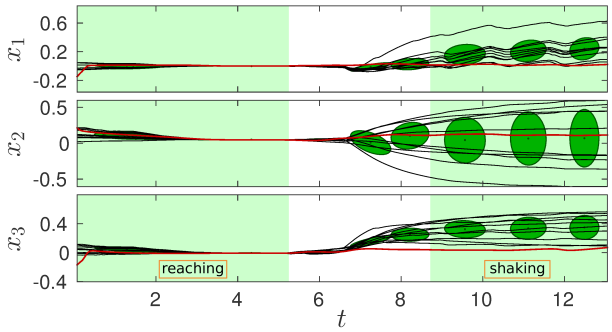


Fig. 4: Left end-effector position (in meters) with respect to the initial bottle coordinate system (prior to the grasp). Black lines represent demonstrations while the red line represents one reproduction of the movement. Ellipses depict the Gaussian components of the model (isocontour of one standard deviation). The shaded areas mark the duration of the reaching and shaking phases. Notice the low variability in position at the end of the reaching movement.

Figures 4–6 show the demonstration data over time[§] (black lines), in operational and configuration spaces, together with the Gaussian components obtained after EM (green ellipses). In addition we also plot the references generated by GMR (red lines), for a new position and orientation of the bottle. In Figures 4 and 5 we see that, during the reach and grasp movement, there is low variability in the demonstrations, both in position and orientation, when the end-effector is touching the bottle ($t \approx 4s$). This is successfully encoded by the model (narrow Gaussians showing low variance), as this aspect of the skill is important for a correct completion of the task. It follows that the synthesized movement (red line) closely matches the demonstrations in the regions of low variability. Note that, after the grasp ($t > 7s$), the variance increases as the end-effectors move away from the initial bottle pose to perform the shaking movement. Figure 6 shows that, from the beginning of the shaking phase ($t \approx 8s$), the shoulder joint (bottom graph) exhibits a consistent oscillatory pattern modeled by 3 Gaussians, which is adequately captured and

[§]Due to space limitations we only plot data corresponding to the left arm.

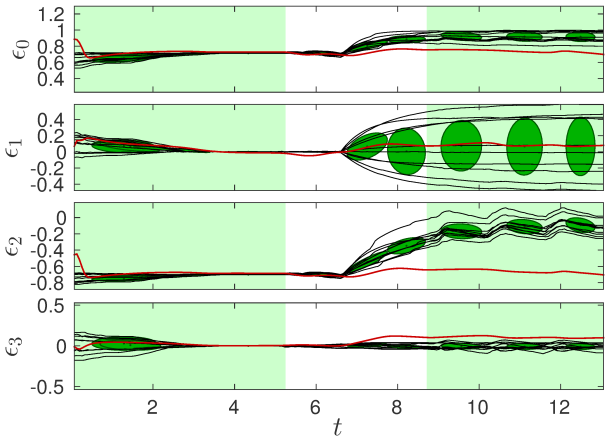


Fig. 5: Left end-effector orientation (as a unit quaternion) with respect to the initial bottle coordinate system (prior to the grasp). Notice the low variability in orientation at the end of the reaching movement.

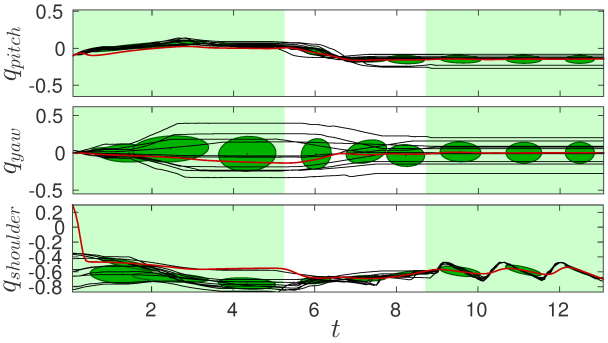


Fig. 6: Waist joints (pitch and yaw) and shoulder joint of the left arm (in radians). Notice the low variability of the shoulder joint during the shaking part of the movement (second shaded area) and how the model captures the shaking pattern encoded in this joint.

synthesized by the model. This contrasts with the other joints of the robot, which do not influence the shaking.

When using the task parameters defined in this section, each set of operators is responsible for a candidate configuration space solution. The weight of each solution is estimated from the demonstrations, based on the variability in the data, and encoded by the different Gaussians as full covariance matrices. In this sense, the approach implements a form of task prioritization based on a *fusion of tasks* (see Section II for a review of prioritization strategies), where full precision matrices act as weights on the candidate solutions. Figure 7 shows that TP-GMM correctly extracted the most relevant configuration space solution according to the requirements of the overall task. Notice how, until $t \approx 5s$, TP-GMM (black line) matches the candidate solution given by the bottle coordinate system (red line). Since the variability encoded

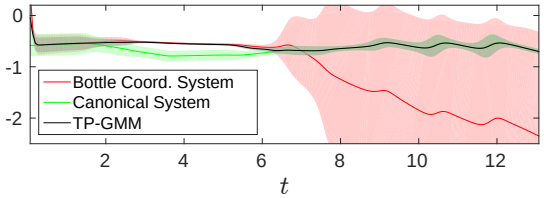


Fig. 7: Shoulder joint angle estimation (radians) from each space (red and green) and the resulting reference after TP-GMM (black). Each estimate has an associated mean (solid line) and variance (light color envelope), learned from demonstrations and synthesized during reproduction.

in this coordinate system is low compared to that of the configuration space (because reaching and grasping is done in operational space), the Gaussian product favors this solution. This is achieved through the linear transformation properties of Gaussians, that allow for both centers and covariance matrices to be locally mapped from operational to configuration space using the proposed linear operators. Similarly, during the shaking phase, after $t \approx 8s$, the reference generated for the shoulder joint matches the solution obtained using the canonical projection operator. This is because the shaking movement results in a consistent oscillatory pattern of shoulder joints, observed during the demonstrations, as seen in Figure 6. These results show that the proposed TP-GMM formulation is a viable solution for encoding task relevant features in both configuration and operational spaces, including orientation.

Finally, Figure 3 shows the two distinct phases of the movement during a reproduction in the real COMAN robot. In this experiment, we used a tray to carry a shaker towards COMAN, where an optical tracking system provided the shaker pose to the robot. We employed the learned TP-GMM to generate joint references at every time step of the reproduction, which were fed to a joint position controller. In the top row of Figure 3, the robot takes into account the operational space constraints as it reaches for the bottle, while in the bottom row, the robot shakes the grasped bottle with rhythmic shoulder movements. In both parts of the movement, the operational and configuration space constraints are properly replicated. Videos are available at `http://joaosilverio.weebly.com/tro` .

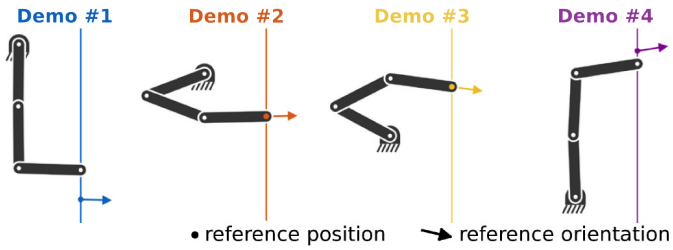
V. Learning Task Prioritization Hierarchies from Demonstrations

Controlling robots often requires the definition of priorities between tasks. For example, when a humanoid is standing and has to manipulate an object, the highest priority should be on keeping balance and, therefore, all degrees of freedom should be assigned to that task and only manipulate when balance is not compromised. Commonly, the way in which tasks are prioritized is defined beforehand by an expert [15], [19]. In contrast, here we propose to learn these priorities from demonstrations and develop a framework to do so. We frame the problem in the context of TP-GMM by formulating the task parameters as candidate hierarchies and subsequently learning how to fuse the different hierarchies from demonstrations. Our approach is hence comprised of two components:

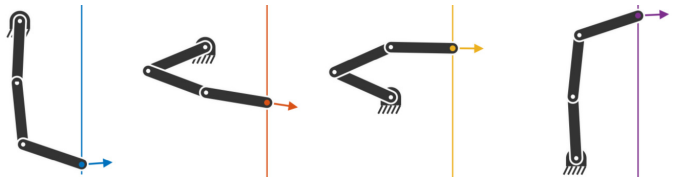
1) **Hierarchy identification (Section V-A):** From a set of candidate task hierarchies, and knowledge of the possible control references for each sub-task, we identify the employed hierarchies based on the variability present in the demonstrations.
2) **Movement synthesis (Section V-B):** A controller is used to reproduce the taught priority behaviors in new situations, through a fusion of candidate hierarchies.

A. Identifying priority hierarchies from demonstrations

We consider joint velocity commands generated from strict hierarchies, where tasks of lower importance are performed in the null space of more important ones, i.e., they are only executed if they do not conflict. In this subsection we show how the prioritization employed during demonstrations can be identified from a set of candidate hierarchies.



(a) Four demonstrations of priority on **orientation**. The robot only fulfills the two tasks when they are both achievable (demonstrations 2 and 3). When they are incompatible, orientation is prioritized (demonstrations 1 and 4).



(b) Four demonstrations of priority on **position**. Once the two tasks become incompatible the robot prioritizes position tracking (demonstrations 1 and 4).

Fig. 8: Planar robot tracking an orientation (arrow) and a vertical position (dot) along a line, with different priorities.

In order to ease the explanation, let us consider an example of a 3-DOF planar robot with position and orientation tracking tasks (Figure 8). We denote the operational space velocities that ensure the tracking of position and orientation references by $\dot{\boldsymbol{x}}_1 = \boldsymbol{K}_p \boldsymbol{e}_p$ and $\dot{\boldsymbol{x}}_2 = \boldsymbol{K}_o \boldsymbol{e}_o$, respectively, where $\boldsymbol{K}_p, \boldsymbol{K}_o$ are positive gains and $\boldsymbol{e}_p = \hat{\boldsymbol{x}}_p - \boldsymbol{x}_p, \boldsymbol{e}_o = \hat{\boldsymbol{x}}_o - \boldsymbol{x}_o$ are the task errors (in principle, any controller that generates a command proportional to the error in the reference state would be a valid choice). These two tasks can be prioritized according to

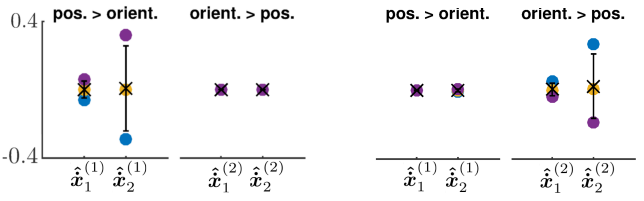
$$\hat{\dot{\boldsymbol{q}}}^{(1)} = \boldsymbol{J}_1^{\dagger}\, \dot{\boldsymbol{x}}_1 + \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger}\, \dot{\boldsymbol{x}}_2 = \underbrace{\begin{bmatrix} \boldsymbol{J}_1^{\dagger} & \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \end{bmatrix}}_{\boldsymbol{A}^{(1)}} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix} \tag{17}$$

or, alternatively,

$$\hat{\dot{\boldsymbol{q}}}^{(2)} = \boldsymbol{J}_2^{\dagger}\, \dot{\boldsymbol{x}}_2 + \boldsymbol{N}_2 \boldsymbol{J}_1^{\dagger}\, \dot{\boldsymbol{x}}_1 = \underbrace{\begin{bmatrix} \boldsymbol{N}_2 \boldsymbol{J}_1^{\dagger} & \boldsymbol{J}_2^{\dagger} \end{bmatrix}}_{\boldsymbol{A}^{(2)}} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}, \tag{18}$$

where $\boldsymbol{J}_1, \boldsymbol{J}_2$ and $\boldsymbol{N}_1, \boldsymbol{N}_2$ are the Jacobians of each task and corresponding null space projection matrices. Figure 8 shows the two different prioritization strategies: priority on orientation (Figure 8a) and priority on position (Figure 8b). Each demonstration is a snapshot of the robot, after satisfying the task space constraints as best as possible. In both scenarios, when the two references can be achieved simultaneously (orange and yellow references), the robot successfully fulfills the two tasks. When the two references are incompatible, the robot prioritizes one over the other, according to the demonstrated hierarchy (blue and purple references).

We propose to treat the linear operators $\{\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}\}$, defined in (17), (18) as two *candidate hierarchies*, which prioritize the tasks differently. The aim is to learn which hierarchy was demonstrated, given observations of the desired operational space velocities $\dot{\boldsymbol{x}}_1, \dot{\boldsymbol{x}}_2$. These are equivalent to the task errors, up to a constant gain. One way to do this is through



(a) Demonstrated **priority on orientation** (see Figure 8a).

(b) Demonstrated **priority on position** (see Figure 8b).

Fig. 9: Operational space velocities generated by each candidate hierarchy for the configurations of the robot in Figure 8. Each colored point corresponds to a demonstration with the same color in Figure 8. The '×' represents the mean of the datapoints, while the bars depict one standard deviation.

the analysis of the task space motion $\hat{\dot{\boldsymbol{x}}}^{(j)} = \boldsymbol{J}\hat{\dot{\boldsymbol{q}}}^{(j)}$ that would result from the application of each candidate hierarchy $j \in \{1, 2\}$, i.e.,

$$\begin{bmatrix} \hat{\dot{\boldsymbol{x}}}_1^{(1)} \\ \hat{\dot{\boldsymbol{x}}}_2^{(1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix} \hat{\dot{\boldsymbol{q}}}^{(1)} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{J}_1^{\dagger} & \boldsymbol{N}_1\boldsymbol{J}_2^{\dagger} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}, \tag{19}$$

for hierarchy $\boldsymbol{A}^{(1)}$, and

$$\begin{bmatrix} \hat{\dot{\boldsymbol{x}}}_1^{(2)} \\ \hat{\dot{\boldsymbol{x}}}_2^{(2)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix} \hat{\dot{\boldsymbol{q}}}^{(2)} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{N}_2\boldsymbol{J}_1^{\dagger} & \boldsymbol{J}_2^{\dagger} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}, \tag{20}$$

for $\boldsymbol{A}^{(2)}$. Here, $\dot{\boldsymbol{x}}_1, \dot{\boldsymbol{x}}_2$ are computed from the demonstrations, given the references and the end-effector position/orientation, and they have the same value in both (19) and (20). Figure 9 shows the result of applying (19), (20) to the examples of Figure 8 (with $\dot{\boldsymbol{x}}_1, \dot{\boldsymbol{x}}_2$ computed from each task error in the depicted configuration). Notice how the datapoints exhibit low variability for the hierarchy that was demonstrated in each case (Figures 9a and 9b). The subspaces $j \in \{1, 2\}$ associated with each candidate hierarchy $\boldsymbol{A}^{(1)}$ and $\boldsymbol{A}^{(2)}$, as computed from (19) and (20), can thus be seen as containing information about the priority constraints that were demonstrated and need to be fulfilled by the robot. We thus propose to exploit the observed variability to assign importance to each candidate hierarchy. For this, we model the distribution of the data in each subspace j with a local GMM, where $[\boldsymbol{\xi}^{\mathcal{I}} \;\; \hat{\dot{\boldsymbol{x}}}^{(j)}] \sim \sum_{i=1}^{K} \pi_i \mathcal{N}\left(\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\right)$ with $\boldsymbol{\xi}^{\mathcal{I}}$ an input. Model estimation is done using EM, as discussed in Section III-B. The covariance matrices $\boldsymbol{\Sigma}_i^{(j)}$ encode the demonstrated variability in the subspace of each candidate hierarchy and they are here exploited to identify priorities. In Section V-B we show how $\boldsymbol{\Sigma}_i^{(j)}$ are transformed into matrix weights for reproducing demonstrated priority behaviors in new situations.

We can easily generalize the principle used in (19) and (20) to N_T arbitrary tasks and $j = 1, \ldots, P$ candidate hierarchies, represented by $\boldsymbol{A}^{(j)}$. Indeed, such generalization takes the form

$$\begin{bmatrix} \hat{\dot{\boldsymbol{x}}}_1^{(j)} \\ \vdots \\ \hat{\dot{\boldsymbol{x}}}_{N_T}^{(j)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_1 \\ \vdots \\ \boldsymbol{J}_{N_T} \end{bmatrix} \boldsymbol{A}^{(j)} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \vdots \\ \dot{\boldsymbol{x}}_{N_T} \end{bmatrix}, \tag{21}$$

from which follows that local datasets are computed with $\boldsymbol{X}_t^{(j)} = \boldsymbol{J}_t\boldsymbol{A}_t^{(j)}\boldsymbol{\xi}_t$, where $\boldsymbol{J}_t = \begin{bmatrix} \boldsymbol{J}_{t,1}^{\mathsf{T}} \; \ldots \; \boldsymbol{J}_{t,N_T}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$ and $\boldsymbol{\xi}_t = \begin{bmatrix} \dot{\boldsymbol{x}}_{t,1}^{\mathsf{T}} \; \ldots \; \dot{\boldsymbol{x}}_{t,N_T}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$. Note that, in this approach, we assume that the kinematic model of the robot is known, i.e., that the Jacobian of each task is available during the demonstrations, and that the set-point of each task is also given.

B. Movement synthesis: fusion of strict hierarchies

The most common approaches for prioritizing tasks are either based on strict hierarchies [28], [29], [31], [33], or soft weighting of tasks [15], [16], [17], [35]. We propose a richer alternative based on a *fusion of strict hierarchies*, gathering the best of the two approaches. For this, we exploit the learned variability in the subspace of each hierarchy, given by full covariance matrices $\boldsymbol{\Sigma}_i^{(j)}, \forall i = 1, \ldots, K, \forall j = 1, \ldots, P$ to propose a controller (dropping time subscripts t for ease of notation)

$$\hat{\dot{\boldsymbol{q}}} = \arg\min_{\dot{\boldsymbol{q}}} \sum_{j=1}^{P} \left(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^{(j)}\right)^{\mathsf{T}} \boldsymbol{\Gamma}^{(j)} \left(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^{(j)}\right), \tag{22}$$

which combines candidate joint space velocities $\dot{\boldsymbol{q}}^{(j)}$ with precision matrices $\boldsymbol{\Gamma}^{(j)}$. We solve the problem (22) in two steps: estimating $\boldsymbol{\Gamma}^{(j)}$ and computing $\hat{\dot{\boldsymbol{q}}}$. For $\boldsymbol{\Gamma}^{(j)}$, GMR is first used to compute the distribution $\hat{\dot{\boldsymbol{x}}}^{(j)} | \boldsymbol{\xi}_t^{\mathcal{I}} \sim \mathcal{N}\left(\boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}^{(j)}\right)$ for a new input at each time step t. The covariance matrix $\boldsymbol{\Sigma}^{(j)}$, which models the importance of hierarchy j for the considered input, is then transformed with task parameters $\boldsymbol{A}^{(j)}$ to compute a precision matrix

$$\boldsymbol{\Gamma}^{(j)} = \left(\boldsymbol{A}^{(j)}\boldsymbol{\Sigma}^{(j)}\boldsymbol{A}^{(j)^{\mathsf{T}}}\right)^{-1}. \tag{23}$$

Note that $\boldsymbol{\Sigma}^{(j)}$ is a squared matrix with the same number of rows and columns as the dimension of $\dot{\boldsymbol{x}}^{(j)}$. Hence $\boldsymbol{A}^{(j)}$ maps the covariance matrices onto joint space. $\boldsymbol{\Gamma}^{(j)}$ are thus precision matrices that reflect the importance of each hierarchy for a given input (e.g., small values of $\boldsymbol{\Sigma}^{(j)}$ will result in high values of $\boldsymbol{\Gamma}^{(j)}$). Then, we compute a candidate joint space velocity for each hierarchy $j = 1, \ldots, P$ as

$$\dot{\boldsymbol{q}}^{(j)} = \boldsymbol{A}^{(j)}\dot{\boldsymbol{x}}^{(j)}, \tag{24}$$

where $\dot{\boldsymbol{x}}^{(j)}$ are the desired operational space velocities for the tasks in hierarchy j. Finally, it can be shown that the solution of (22) corresponds to a product of P Gaussians with mean $\dot{\boldsymbol{q}}^{(j)}$ and precision matrix $\boldsymbol{\Gamma}^{(j)}$, i.e.,

$$\hat{\dot{\boldsymbol{q}}} = \left(\sum_{j=1}^{P} \boldsymbol{\Gamma}^{(j)}\right)^{-1} \sum_{j=1}^{P} \boldsymbol{\Gamma}^{(j)}\dot{\boldsymbol{q}}^{(j)}, \tag{25}$$

similarly as in (2). The solution $\hat{\dot{\boldsymbol{q}}}$ is a reference joint velocity, which can be used to control the robot through a velocity or a position controller (in which case it should be integrated numerically).

Note that the complete procedure, from estimating local models based on (21) to obtaining a solution (25), is a modified version of TP-GMM, where the solutions for each candidate hierarchy $\dot{\boldsymbol{q}}^{(j)}$ replace the predicted centers $\hat{\boldsymbol{\mu}}_{t,i}^{(j)}$ in the original formulation (2) and $\boldsymbol{b}^{(j)} = \boldsymbol{0}$. The complete approach described in Sections V-A and V-B is summarized in Algorithm 2.

Algorithm 2 Learning task prioritization hierarchies from demonstrations

Initialization

Select a set of potentially relevant candidate hierarchies $\boldsymbol{A}^{(j)}$
Collect demonstrations

Model training

Compute $\boldsymbol{X}^{(j)}$ for each candidate hierarchy $\boldsymbol{A}^{(j)}$ with (21)
Estimate model $\left\{\pi_i, \{\boldsymbol{\mu}_i^{(j)}, \boldsymbol{\Sigma}_i^{(j)}\}_{j=1}^{P}\right\}_{i=1}^{K}$ using EM

Movement synthesis

for $t = 1, \ldots, N$ **do**
  Obtain input $\boldsymbol{\xi}_t^{\mathcal{I}}$ at time step t
  for $j = 1, \ldots, P$ **do**
    Update task parameters $\boldsymbol{A}^{(j)}$ with Jacobians and null space matrices at time step t
    Compute $\boldsymbol{\Gamma}^{(j)}$ using GMR and (23)
    Update desired task space velocities $\dot{\boldsymbol{x}}^{(j)}$ for the tasks in hierarchy j
    Compute candidate solution $\dot{\boldsymbol{q}}^{(j)}$ using (24)
  end for
  Compute velocity control command $\hat{\dot{\boldsymbol{q}}}$ using (25)
end for

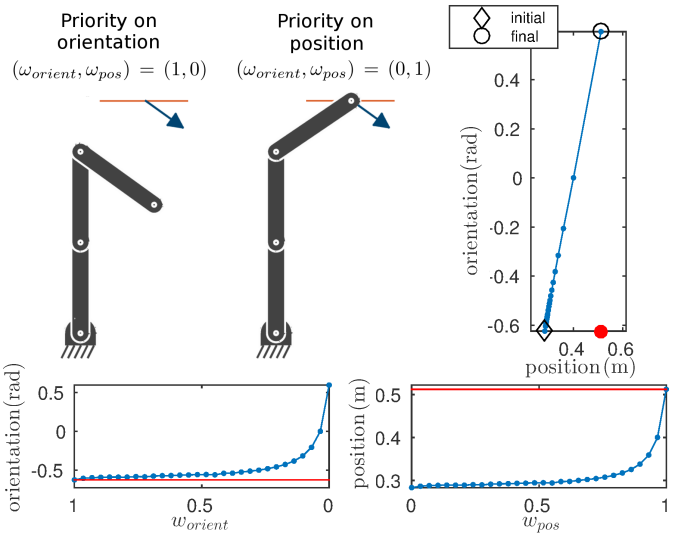


Fig. 10: End-effector behavior for varying hierarchy weights. **Top-left:** Planar robot fulfilling the two tasks separately. **Top-right:** End-effector state in the position-orientation space for the considered weights (blue line). **Bottom:** End-effector orientation and vertical position as weights vary. Note the inverted scales on the horizontal axes (the experiment starts with $(1, 0)$ and ends with $(0, 1)$). Red lines and point indicate the references.

C. Experiment: Fusion of hierarchies and behavior during transitions

An important aspect to consider when controlling priorities is the behavior of the robot when priorities change. Before testing the learning capabilities of our approach, we study the effect of varying the weights of each candidate hierarchy, when employing the controller proposed in Section V-B. For simplicity of analysis, we consider the case of a planar robot with two tasks. The tasks are to track a vertical position reference with the end-effector and to point downwards. The references are chosen such that the two tasks can be fulfilled individually but not simultaneously (Figure 10, *top-left*). Two strict hierarchies are considered, corresponding to an accurate tracking of either position or orientation as first priority, and orientation or position as secondary. We define two scalar

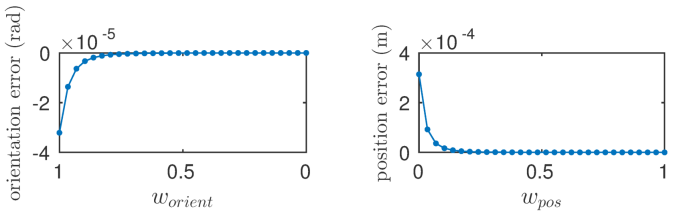


Fig. 11: Performance of the robot in tracking position and orientation references with varying weights when both tasks can be achieved.

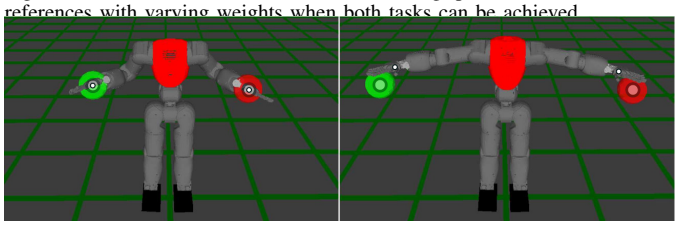


Fig. 12: Illustration of the waist joint prioritization problem. **Left:** Both targets are within the reachable workspace of both arms, so the robot can successfully reach the centers of the spheres with the palm of its hands. **Right:** References become simultaneously unreachable, so the robot cannot reach any of the two. The small circles depict the tool center point of each arm, while the large circles represent the centers of the references.

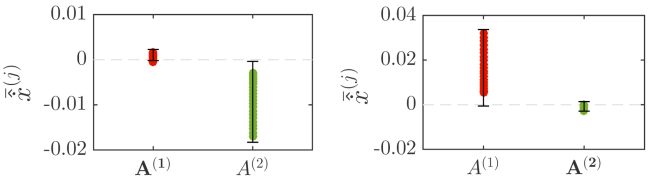
weights w_{pos} and w_{orient}, which represent the importance of each hierarchy, with the subscript denoting the highest priority task. With these weights, we manually define precision matrices $\boldsymbol{\Gamma}_{\text{pos}} = w_{\text{pos}}\boldsymbol{I}$ and $\boldsymbol{\Gamma}_{\text{orient}} = w_{\text{orient}}\boldsymbol{I}$, where $\boldsymbol{I}$ is a 2-dimensional identity matrix. Figure 10 shows the results for the case in which the robot transits from priority on orientation to priority on position, through a continuous variation of the weights. We observe that, as different combinations of weights are applied by the controller, a smooth transition occurs between the two tasks¶. These results show that the proposed controller can handle situations where priorities change, which can occur when demonstrated hierarchies vary over time or according to different contexts.

Another case that is worth investigating is the behavior of the system when both tasks are achievable. In this situation, one would expect that any combination of the weights would result in a successful completion of both tasks. In order to test how our approach handles this scenario, we decreased the vertical position reference so that both tasks can be fulfilled simultaneously. We set the robot to an initial configuration $\boldsymbol{q}_{\text{init}} = [\frac{\pi}{2}+0.5,\ -1,\ -\frac{\pi}{2}+0.5]$. For every new weight pair, its starting configuration is that of the previous pair. Figure 11 shows the tracking performance. We can see that both position and orientation were tracked with negligible errors. This shows that the approach converges to a proper solution and maintains it when the weights are modified. In other words, once the robot converged to a configuration that fulfills both tasks with the initial weights $(w_{\text{orient}}, w_{\text{pos}}) = (1, 0)$, further variations of the weights did not affect the solution, as one would expect.

D. Experiment: Learning priorities with the COMAN robot

We now test the hierarchy identification and synthesis capabilities of the approach with the humanoid robot COMAN.

¶Note that the transition is non-linear, unlike the variation of the weights w_{pos}, w_{orient}. This is because the Jacobians of each task, involved in (23), change non-linearly with the robot configuration $\boldsymbol{q}$.



(a) Demonstrated **left arm priority**. (b) Demonstrated **right arm priority**.

Fig. 13: Task space velocities generated by each candidate hierarchy given the demonstration data. Each entry $\boldsymbol{A}^{(j)}$ contains the means of several observations $\hat{\boldsymbol{x}}^{(j)}$, computed across the dimensions of the tasks. The error bars correspond to two standard deviations of the data. Note the low variability in the datapoints corresponding to the demonstrated hierarchies.

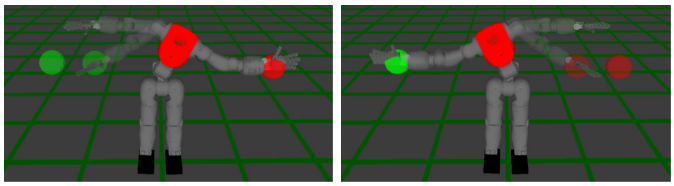


Fig. 14: Movement synthesis using the learned left and right arm priority models. **Left:** The left reference is always tracked, while the right reference is only tracked when reachable. **Right:** The right reference is prioritized.

Since the kinematic chains of the humanoid arms share the waist joints, incompatibilities often occur during bimanual manipulation when the tasks of the two arms are too distant (Figure 12). In this experiment we use the proposed approach to teach the robot which arm has priority over the other. If correctly learned, the robot will prioritize one of the references when the two are incompatible and closely track both when reachable. We consider $P = 2$ candidate hierarchies, corresponding to the two possible combinations of priorities, i.e., priority on left arm with the right arm as secondary and vice-versa. We denote the Jacobians of each arm (from the waist link) by $\boldsymbol{J}_L$, $\boldsymbol{J}_R$, with corresponding pseudoinverses $\boldsymbol{J}_L^\dagger$, $\boldsymbol{J}_R^\dagger$ and null space projection matrices $\boldsymbol{N}_L$, $\boldsymbol{N}_R$, yielding the task parameters

$$\boldsymbol{A}^{(1)} = \begin{bmatrix} \boldsymbol{J}_L^\dagger & \boldsymbol{N}_L \boldsymbol{J}_R^\dagger \end{bmatrix}, \qquad \boldsymbol{A}^{(2)} = \begin{bmatrix} \boldsymbol{N}_R \boldsymbol{J}_L^\dagger & \boldsymbol{J}_R^\dagger \end{bmatrix}, \tag{26}$$

and the desired task space velocities for each hierarchy

$$\dot{\boldsymbol{x}}^{(1)} = \dot{\boldsymbol{x}}^{(2)} = \begin{bmatrix} \hat{\boldsymbol{x}}_L - \boldsymbol{x}_L \\ \hat{\boldsymbol{x}}_R - \boldsymbol{x}_R \end{bmatrix}, \tag{27}$$

where $\hat{\boldsymbol{x}}_L$ and $\hat{\boldsymbol{x}}_R$ denote left and right reference positions. For model training, we consider datapoints of the form $\boldsymbol{\xi}_t = \begin{bmatrix} \dot{\boldsymbol{x}}_{L,t}^\mathsf{T} & \dot{\boldsymbol{x}}_{R,t}^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^6$, where $\dot{\boldsymbol{x}}_{L,t} = \hat{\boldsymbol{x}}_{L,t} - \boldsymbol{x}_{L,t}$ and $\dot{\boldsymbol{x}}_{R,t} = \hat{\boldsymbol{x}}_{R,t} - \boldsymbol{x}_{R,t}$ are the desired left and right end-effector velocities during demonstrations. Equation (21) is used to compute the local datasets $\boldsymbol{X}^{(j)}$. We choose $K = 1$, i.e., we want to have a constant prioritization structure throughout the task. This is to show that we can already encode complex behaviors with a single Gaussian. More complex structures can be easily considered by adding more components (e.g., to teach input dependent priorities).

Demonstrations of the priority behaviors are given by implementing an inverse kinematics controller with the desired hierarchies in the simulated robot and making it track moving

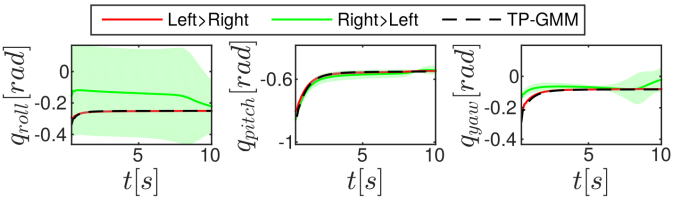


Fig. 15: Waist joint angles given by each candidate hierarchy. The envelope around the lines represents one standard deviation of the variance encoded in each hierarchy. TP-GMM extracts the solution with the lowest variability.

$\boldsymbol{A}^{(1)}$	$\boldsymbol{A}^{(2)}$	$\boldsymbol{A}^{(3)}$	$\boldsymbol{A}^{(4)}$	$\boldsymbol{A}^{(5)}$	$\boldsymbol{A}^{(6)}$
$b>p>o$	$b>o>p$	$p>o>b$	$p>b>o$	$o>p>b$	$o>b>p$

TABLE IV: Hierarchies used in Section V-E.

references[‖]. We generate demonstrations of two different types of priority and analyzed the resulting datasets. Figure 13 shows the mean of $\hat{\boldsymbol{x}}^{(j)}$, generated by each candidate hierarchy, computed from (21) with task parameters (26). As one would expect from Section V-A, we observe a distribution of datapoints with low variability for the hierarchy that was demonstrated, with priority either on the left arm (Figure 13a) or on the right arm (Figure 13b). Low variability results in high precision matrices $\boldsymbol{\Gamma}^{(j)}$, which in turn, during movement synthesis, result in higher weights to favor the corresponding hierarchy.

We apply the learned models (left and right arm priority) to reach fixed points on the left and right sides of the robot, with the results depicted in Figure 14. As expected, in both cases, we observe that the arm which had the highest priority during demonstrations always fulfills the task, while the other only does it when the reference is reachable. Figure 1 shows snapshots of the same behaviors in the real COMAN platform, where we used an optical tracking system to provide moving references to each arm. The robot closely tracked the reference on the side that had the highest priority, while doing its best to track the secondary reference. The computations in rows 2–9 of Algorithm 2 (computation at each time step) took on average $1.15ms$, using MATLAB in a laptop with an Intel Core i5-3230M, 2.60GHz $\times$ 4 processor. Videos of the experiment can be found at `http://joaosilverio.weebly.com/tro`.

In this framework, similarly to the experiment in Section IV-D, each candidate hierarchy provides one possible configuration space solution. Figure 15 shows the solution given by each candidate hierarchy for the waist roll, pitch and yaw joints, while tracking the left object with the highest priority. We see that the solution given by the hierarchy that encodes priority on the left arm (red line) has much lower variability than the one that prioritizes tracking with the right arm (green line). This leads to an automatic identification of this prioritization as being important, resulting in a proper reproduction of priorities.

E. Experiment: Learning priorities with more than two tasks

In this experiment we consider a loco-manipulation task with the new Centauro robot from IIT [13]. Centauro is a

[‖] Alternatively, kinesthetic teaching or optical tracking of movements from humans could also be used by recording the tracking errors as well as the kinematic data of the demonstrator (required to compute the Jacobians).

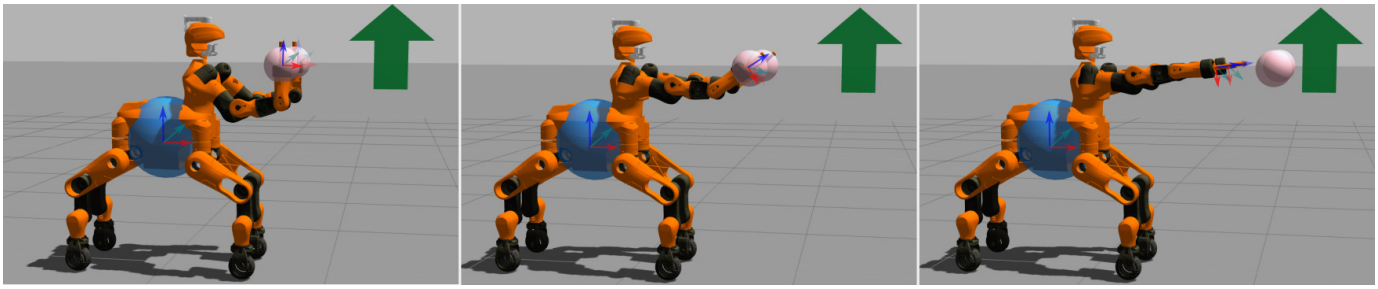
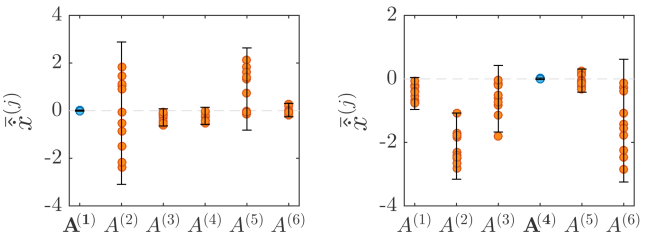


Fig. 16: The Centauro robot tracks different references, prioritizing (by decreasing order of importance) base position, end-effector positions and end-effector orientations. The blue and pink spheres represent the desired base and end-effectors positions, while the arrow denotes the desired end-effector pointing direction. **Left:** All tasks can be fulfilled. **Center:** The pelvis reference is moved backwards, thus the robot performs the orientation task as best as it can but not completely. **Right:** The end-effector position references are moved forward and both end-effector position and orientation are now only partially fulfilled.



(a) **Demonstrated:** $b > p > o$. (b) **Demonstrated:** $p > b > o$.

Fig. 17: Datasets from Centauro priority demonstrations. Each entry $\boldsymbol{A}^{(j)}$ contains 12 points (the means of $\hat{\dot{\boldsymbol{x}}}^{(j)}$ for each demonstration).

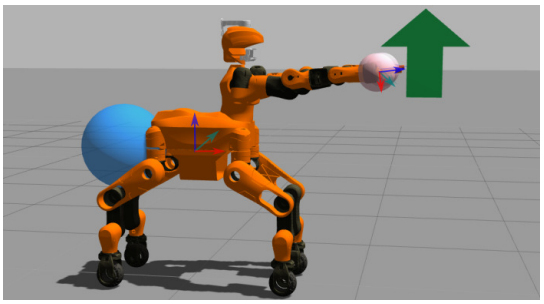


Fig. 18: In this example, the Centauro robot prioritizes end-effector positions, followed by the position of the base and, finally, the end-effector orientations.

four-legged robot with wheels and a 15-DOF humanoid torso. In order to simplify the locomotion problem, we constrain the robot to move using only the wheels, becoming, in essence, a mobile platform with a humanoid torso. We consider three tasks that the robot can execute at the same time, corresponding to the control of:

- the *position* of the robot's *base*, assumed to be the pelvis link, with associated Jacobian and null space projection matrices $\boldsymbol{J}_b, \boldsymbol{N}_b$,
- the *positions* of *both end-effectors*, with matrices $\boldsymbol{J}_p, \boldsymbol{N}_p$,
- the *orientations* of *both end-effectors*, with $\boldsymbol{J}_o$,$\boldsymbol{N}_o$.

The possible arrangements of these three tasks yield 6 candidate hierarchies (see Table IV where we used the Jacobian subscripts to denote each task). In many contexts, one may know in advance that the robot will only employ a subset of the total available hierarchies (e.g., one may assume that the pelvis position is not an important sub-task). We here consider all possible combinations with $P = 6$, with the aim of showing the potential of the approach to resolve tasks with a complete set of candidate hierarchies. In this experiment, we control a total of 19 joints: 4 wheels and 15 upper body joints (7 per arm and 1 torso joint).

Using the Gazebo simulator, we collect 12 demonstrations of the robot employing the hierarchy $\boldsymbol{A}^{(1)} = [\boldsymbol{J}_b^\dagger \quad \boldsymbol{N}_b\boldsymbol{J}_p^\dagger \quad \boldsymbol{N}_b\boldsymbol{N}_p\boldsymbol{J}_o^\dagger]$, which controls the base position with the highest priority, followed by the end-effector positions and, finally, orientations. Each demonstration uses different combinations of the position references (base and end-effectors), while keeping the orientation reference the same, in this case the quaternion $\hat{\boldsymbol{\epsilon}} = [0 \ 0 \ 0 \ 1]^\mathsf{T}$. Figure 16 shows 3 demonstration examples, for different values of the references to show the priority behaviors. The demonstration data is stored when the robot is at a configuration where each task is fulfilled as best as possible, according to the hierarchy. Each demonstration results in P vectors $\hat{\dot{\boldsymbol{x}}}^{(j)}$, as computed from (21).

Figure 17a shows the mean of the resulting vectors for each $j = 1, \ldots, P$, with error bars denoting two standard deviations. Notice that the demonstrated hierarchy generated the lowest variance. This confirms our proposition from Section V-A, validating the approach for the extraction of hierarchies with more than two tasks. The reproduction of the learned priorities, for the same and different references, is shown in the video that can be found at `http://joaosilverio.weebly.com/tro`. Also note that, in Fig. 17a, we take the mean of $\hat{\dot{\boldsymbol{x}}}^{(j)}$ only for visualization purposes. The reproduction of priority behaviors is done using (25), which takes into account the variability of each dimension separately. For this 6-hierarchy experiment, the operations in rows 2–9 of Algorithm 2 (computation at each time step) took on average $7.35ms$ per control cycle, in a computer with an Intel Core i7-6700, $3.40\text{GHz} \times 8$ processor using a C++ (unoptimized) implementation based on the Eigen library for linear algebra computations. The increase in computational time with respect to the experiment reported in Section V-D is due to a higher number of hierarchies and controlled degrees of freedom (we considered the 48-dimensional whole-body Jacobian), which increases the cost of the inversions in (23) and (25).

In order to show that the approach can successfully extract any demonstrated hierarchy, we provide 12 new demonstrations of a different hierarchy, in this case $\boldsymbol{A}^{(4)} = [\boldsymbol{N}_p\boldsymbol{J}_b^\dagger \quad \boldsymbol{J}_p^\dagger \quad \boldsymbol{N}_p\boldsymbol{N}_b\boldsymbol{J}_o^\dagger]$, where the end-effector positions are prioritized, followed by the pelvis position and end-effector orientations. Figure 18 shows an example of such hierarchy. In Fig. 17b, we see that the demonstrated hierarchy

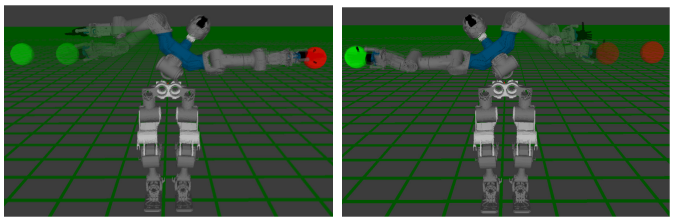


Fig. 19: Movement synthesis with the WALK-MAN robot using the left and right priority models that were learned with COMAN.

is properly extracted, since the data show low variability for $\boldsymbol{A}^{(4)}$. The accompanying video shows the reproduction of this hierarchy, highlighting how it differs from $\boldsymbol{A}^{(1)}$.

F. Experiment: Transferring task priorities between robots

In this experiment we test the transfer of priority models across different robots. We use the model learned with COMAN to synthesize an equivalent prioritization behavior in WALK-MAN [44] that has different kinematic parameters. WALK-MAN is a humanoid larger than COMAN, whose waist joints are also shared between the arms. Figure 19 shows that the model acquired using COMAN is successful in generalizing the learned task priorities in WALK-MAN. This stems from two facts: i) the considered tasks, and associated weights, are in operational space, whose dimension is the same for both robots and ii) the robots have the same kinematic chains structure, i.e., waist joints are shared by both arms. This opens up an interesting strategy to cope with the correspondence problem in imitation learning. In particular, our approach can potentially be used by the robot to visually observe a task prioritization behavior demonstrated by the user without considering a direct mapping of the kinematics.

VI. Discussion

A. Learning operational and configuration space constraints

Our results in Section IV showed that, with Jacobian-based task parameters, the robot can take into account both configuration and operational space constraints, including orientation. Learning controllers for complex bimanual manipulation goes beyond operational space, and one must also care about the configuration space to achieve natural and efficient movements. The shaking experiment is an example of task that is best modeled by taking into account the configuration space constraints, but the range of possibly interesting situations is vast (e.g., communicative gestures, self-collision avoidance). Although highly relevant, this problem is rarely covered in the literature of motor skill learning, especially in bimanual cases.

One potential shortcoming of our approach is that it relies on the variability of the demonstrated movements to extract the importance of each space. In some cases, such task variations may not be straightforward to demonstrate (e.g., unexperienced demonstrators may not demonstrate sufficient variability). One possible way to cope with this issue could be to combine TP-GMM with more interactive and incremental teaching approaches, which would allow the robot to refine its task model at runtime by testing task variations in the different spaces based on exploration, interaction and teacher feedback.

In a different direction, we plan to study how the proposed approach can be combined with Riemannian formulations for learning end-effector poses [45]. While here we approximate the geometry of the unit quaternion space, such formulations may allow for a more precise encoding of orientation.

Sections IV and V both considered the combination of sub-spaces as a fusion problem with full precisions matrices, which has a richer structure than a combination of sub-spaces through scalar weights. In particular, they both allow a parallel organization of tasks by exploiting the sparsity of the sub-space constraints. Although better than a superposition through scalar weights, such organization still cannot guarantee that all sub-space constraints can be fulfilled when these constraints are not compatible. Currently, the robot finds a trade-off without evaluating whether this solution is satisfactory or whether sub-spaces are too much in conflict to be fused properly. It would be interesting to investigate in future work if such cases could be detected and how the robot should cope with highly conflicting constraints.

B. Learning task prioritization hierarchies

By extending the notion of candidate coordinate systems to candidate task hierarchies, we showed that TP-GMM could be used to learn priority hierarchies.

One potential shortcoming is that the set of possible task hierarchies needs to be set beforehand. To a certain extent, one can take advantage of expert knowledge to only use a subset of all possible candidate hierarchies. However, even though one is not required to provide all possible hierarchies for the considered tasks, one should be careful not to over-define this set, because the more candidate hierarchies are available, the more demonstrations will be required to extract invariant features through statistics. A potential research direction to leverage this issue could be to study ways of learning sets of hierarchies characterizing specific domains of activity (e.g., bimanual manipulation, walking). Then, one could take advantage of the prior knowledge about the skill to be demonstrated to select the set accordingly.

Another potential drawback is the need to define the set-points that the robot may be tracking. Note that this is a common hypothesis in approaches extracting task prioritization, see e.g., [31], [32]. While this may not pose a problem in simpler scenarios (e.g., the targets do not move), the difficulty increases in highly dynamic environments. One intuitive solution to overcome this limitation could be to split the demonstration of the task in two phases (an idea exploited in [28] albeit in a more simple case). In one phase, each individual sub-task could be demonstrated (e.g., pelvis trajectory or end-effector movements with respect to objects) and encoded in separate models. In the second phase, the environment could be simplified (e.g., fixing references) and the priorities demonstrated. Equations (24), (25) ensure a flexible solution at this level, since the control commands $\dot{\boldsymbol{x}}^{(j)}$ of each sub-task in (24) may be generated from a model that is trained independently from the priority weights used in (25).

VII. Conclusions and Future Work

We presented a framework for human-robot bimanual skill transfer based on Task-Parameterized Gaussian Mixture Models. We introduced different parameterizations allowing us to consider a wide range of learning problems related to manipulation in humanoids, namely the combination of constraints between operational and configuration spaces, and the learning of task priority hierarchies. The approach was validated in three different scenarios. We first demonstrated, in a bimanual shaking task with the COMAN robot, that the approach can be used to consider constraints in operational and configuration spaces simultaneously, which permits the reproduction of motion patterns from both spaces during movement synthesis. Then, in a bimanual reaching experiment with COMAN, we showed that by providing different candidate hierarchies to describe an observed movement, the robot is able to determine from statistics which prioritization is relevant for the task. In this context, we also demonstrated that the proposed formulation could be employed to transfer prioritization behaviors between humanoids. Finally, in a loco-manipulation scenario with the Centauro robot, we showed that our framework can be used to learn the prioritization of more than two simultaneous tasks from demonstrations.

Future work will investigate how other types of constraints could be formulated for torque-controlled robots [46], as opposed to velocity controllers considered in this article. A growing number of robots can be controlled in torque, thus it is relevant for learning approaches to account for this fact.

We also plan to investigate if the proposed approach could be combined with other optimization-based techniques, such as CMA-ES [16], [17], in order to refine the learned priorities. This would allow us to consider cost functions that are harder to relate to operational space control, such as energy efficiency.

A final research direction concerns the application of task prioritization with TP-GMM to more complex whole-body control scenarios. One possible avenue could be that of learning whole body motion behaviors from human demonstrations, including prioritization skills involving center of mass and contact points with the environment (e.g., to learn natural ways of coping with perturbations while standing).

References

[1] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation – a survey," *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340 – 1353, 2012.

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

[3] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[4] J. Umlauft, D. Sieber, and S. Hirche, "Dynamic movement primitives for cooperative manipulation and synchronized motions," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, May-June 2014, pp. 766–771.

[5] R. Lioutikov, O. Kroemer, J. Peters, and G. Maeda, "Learning manipulation by sequencing motor primitives with a two-armed robot," in *Proc. Intl Conf. on Intelligent Autonomous Systems (IAS)*, ser. Advances in Intelligent Systems and Computing, vol. 302. Springer, 2014.

[6] N. Likar, B. Nemec, L. Zlajpah, S. Ando, and A. Ude, "Adaptation of bimanual assembly tasks using iterative learning framework," in *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, Seoul, South Korea, November 2015, pp. 771–776.

[7] A. L. P. Ureche and A. Billard, "Encoding bi-manual coordination patterns from human demonstrations," in *Proc. ACM/IEEE Intl Conf. on Human-Robot Interaction (HRI)*, Bielefeld, Germany, March 2014, pp. 264–265.

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

[9] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell, "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September–October 2015, pp. 464–470.

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

[11] S. Calinon and A. G. Billard, "Statistical learning by imitation of competing constraints in joint space and task space," *Advanced Robotics*, vol. 23, no. 15, pp. 2059–2076, 2009.

[12] N. G. Tsagarakis, S. Morfey, G. A. Medrano-Cerda, Z. Li, and D. G. Caldwell, "COMpliant huMANoid COMAN: Optimal joint stiffness tuning for modal frequency control." in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 673–678.

[13] L. Baccelliere, N. Kashiri, L. Muratore, A. Laurenzi, M. Kamedula, A. Margan, S. Cordasco, J. Malzahn, and N. G. Tsagarakis, "Development of a human size and strength compliant bi-manual platform for realistic heavy manipulation tasks," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 5594–5601.

[14] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, no. 25, pp. 328–373, 2013.

[15] F. L. Moro, M. Gienger, A. Goswami, and N. G. Tsagarakis, "An attractor-based whole-body motion control (WBMC) system for humanoid robots," in *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, Atlanta, GA, USA, October 2013, pp. 42–49.

[16] N. Dehio, R. F. Reinhart, and J. J. Steil, "Multiple task optimization with a mixture of controllers for motion generation," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015, pp. 6416–6421.

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

[18] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[19] L. Sentis and O. Khatib, "Control of Free-Floating Humanoid Robots Through Task Prioritization," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Barcelona, Spain, April 2005, pp. 1718–1723.

[20] C. Ott, A. Dietrich, and A. Albu-Schffer, "Prioritized multi-task compliance control of redundant manipulators," *Automatica*, vol. 53, pp. 416–423, 2015.

[21] J. Park, Y. Choi, W. K. Chung, and Y. Youm, "Multiple tasks kinematics using weighted pseudo-inverse for kinematically redundant manipulators," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Seoul, South Korea, May 2001, pp. 4041–4047.

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

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

[24] B. Morris, M. J. Powell, and A. D. Ames, "Sufficient conditions for the Lipschitz continuity of QP-based multi-objective control of humanoid robots," in *Proc. IEEE Conf. on Decision and Control (CDC)*, Firenze, Italy, December 2013, pp. 2920–2926.

[25] A. Dietrich, A. Albu-Schäffer, and G. Hirzinger, "On continuous null space projections for torque-based, hierarchical, multi-objective manipulation," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, St. Paul, MN, USA, May 2012, pp. 2978–2985.

[26] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, no. 1, pp. 17–31, Jan. 2016.

[27] N. Dehio, D. Kubus, and J. J. Steil, "Continuously shaping projections and operational space tasks," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 2018, pp. 5995–6002.

[28] 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*, vol. 2, pp. 56–81, 2013.

[29] 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)*, Seattle, WA, USA, May 2015, pp. 3570–3575.

[30] S. An and D. Lee, "Prioritized inverse kinematics with multiple task definitions," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015, pp. 1423–1430.

[31] C. Towell, M. Howard, and S. Vijayakumar, "Learning nullspace policies," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010, pp. 241–248.

[32] H.-C. Lin, M. Howard, and S. Vijayakumar, "Learning null space projections," in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015, pp. 2613–2619.

[33] S. Hak, N. Mansard, O. Stasse, and J. P. Laumond, "Reverse control for humanoid robot task recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 6, pp. 1524–1537, December 2012.

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

[35] ——, "Variance modulated task prioritization in whole-body control," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, September 2015, pp. 3944–3949.

[36] A. Paraschos, R. Lioutikov, J. Peters, and G. Neumann, "Probabilistic prioritization of movement primitives," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 4, pp. 2294–2301, October 2017.

[37] 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)*, Hong Kong, China, May-June 2014, pp. 3339–3344.

[38] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, "Learning physical collaborative robot behaviors from human demonstrations," *IEEE Trans. on Robotics*, vol. 32, no. 3, pp. 513–527, 2016.

[39] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.

[40] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, September 2011, pp. 365–371.

[41] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell, "Bimanual skill learning with pose and joint space constraints," in *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, Beijing, China, 2018, pp. 153–159.

[42] J. Lee, A. Ajoudani, E. M. Hoffman, A. Rocchi, A. Settimi, M. Ferrati, A. Bicchi, N. G. Tsagarakis, and D. G. Caldwell, "Upper-body impedance control with variable stiffness for a door opening task," in *Proc. IEEE-RAS Intl Conf. on Humanoid Robots (Humanoids)*, Madrid, Spain, November 2014, pp. 713–719.

[43] C. Y. Chiu, S. P. Chao, M. Y. Wu, and S. N. Yang, "Content-based retrieval for human motion data," *Visual Communication and Image Representation*, vol. 15, pp. 446–466, 2004.

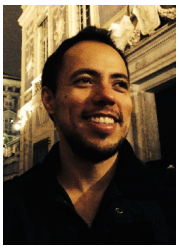
[44] N. G. Tsagarakis, D. G. Caldwell, F. Negrello, W. Choi, L. Baccelliere, V. Loc, J. Noorden, L. Muratore, A. Margan, A. Cardellino, L. Natale, E. Mingo Hoffman, H. Dallali, N. Kashiri, J. Malzahn, J. Lee, P. Kryczka, D. Kanoulas, M. Garabini, M. Catalano, M. Ferrati, V. Varricchio, L. Pallottino, C. Pavan, A. Bicchi, A. Settimi, A. Rocchi, and A. Ajoudani, "WALK-MAN: A High-Performance Humanoid Platform for Realistic Environments," *Journal of Field Robotics*, 2017.

[45] M. J. A. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 3, pp. 1240–1247, June 2017.

[46] J. Silvério, Y. Huang, L. Rozo, S. Calinon, and D. G. Caldwell, "Probabilistic learning of torque controllers from kinematic and force constraints," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Madrid, Spain, October 2018, pp. 6552–6559.

João Silvério is a Postdoctoral Researcher at the Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT) since June 2017. He received his M.Sc in Electrical and Computer Engineering (2011) from Instituto Superior Técnico (Lisbon, Portugal) and Ph.D in Robotics (2017) from the University of Genova (Italy) and IIT. Before his PhD, he also carried out research at EPFL's Biorobotics Laboratory in 2010 and Learning Systems and Algorithms Laboratory in 2013. He is interested in robot programming by demonstration, in particular algorithms for learning complex bimanual skills. Webpage: `http://joaosilverio.eu`

Sylvain Calinon is a Senior Researcher at the Idiap Research Institute. He is also a lecturer at the Ecole Polytechnique Fédérale de Lausanne (EPFL), and an external collaborator at the Department of Advanced Robotics (ADVR), Italian Institute of Technology (IIT). From 2009 to 2014, he was a Team Leader at ADVR, IIT. From 2007 to 2009, he was a Postdoc at the Learning Algorithms and Systems Laboratory, EPFL, where he obtained his PhD in 2007. He is the author of 100+ publications in robot learning and human-robot interaction, with recognition including Best Paper Awards in the journal of Intelligent Service Robotics (2017) and at IEEE Ro-Man'2007, and Best Paper Award Finalist at ICRA'2016, ICIRA'2015, IROS'2013 and Humanoids'2009. He currently serves as Associate Editor in IEEE Transactions on Robotics (T-RO) and IEEE Robotics and Automation Letters (RA-L). Webpage: `http://calinon.ch`



Leonel Rozo is a Team Leader at the Department of Advanced Robotics (ADVR), Istituto Italiano di Tecnologia since 2016. He was a postdoctoral researcher at the same institution from 2013 to 2016. He received his B.Sc on Mechatronics Engineering from the "Nueva Granada" Military University (Colombia, 2005), his M.Sc in Automatic Control and Robotics (2007), and Ph.D in Robotics (2013) from the Polytechnical University of Catalonia (Barcelona, Spain). From 2007 to 2012 he carried out his research on force-based manipulation tasks learning at the Institut de Robòtica i Informàtica Industrial (CSIC-UPC). His research interests cover robot programming by demonstration, physical human-robot interaction, machine learning and optimal control for robotics. Personal webpage: `http://leonelrozo.weebly.com`

Darwin G. Caldwell, FREng is Deputy Director of the Italian Institute of Technology (IIT), and Director of the Dept. of Advanced Robotics at IIT. He is or has been an Honorary Professor at the Universities of Manchester, Sheffield, Bangor, Kings College London and Tianjin University, China. His research interests include; humanoid and quadrupedal robotics (iCub, cCub, COMAN, WalkMan, HyQ, HyQ2Max, HalfMan, COMAN+), innovative actuators, haptics and force augmentation exoskeletons, medical, rehabilitation and assistive robotic technologies, dexterous manipulators. He is the author or co-author of over 500 academic papers, 20+ patents, and has received awards and nominations from many international journals and conferences. Caldwell has been chair of the IEEE Robotics and Automation Chapter (UKRI), a past co-chair of the IEE (IET) Robotics and Mechatronics PN, Secretary of the IEEE/ASME Trans. on Mechatronics, Editor for Frontiers in Robotics and AI, on the editorial board of the International Journal of Social Robotics and Industrial Robot and on the Advisory Board of Science Robotics. In 2015 he was elected a Fellow of the Royal Academy of Engineering.