FP7-600716

Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

D4.3
Learning of the prioritization policies.

Editor(s)	Jan Peters[1,2] and Elmar Rueckert[1]
Responsible Partner	TUDA
Affiliations	[1] Intelligent Autonomous Systems Lab, Technische Universität Darmstadt, 64289 Darmstadt, Germany. [2] Robot Learning Group, Max-Planck Institute for Intelligent Systems, Tuebingen, Germany.
Status-Version:	Draft-1.0
Date:	Feb. 28, 2017
EC Distribution:	Consortium
Project Number:	600716
Project Title:	Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

Title of Deliverable:	Learning of the prioritization policies.
Date of delivery to the EC:	28/2/2017

Workpackage responsible for the Deliverable	WP4
Editor(s):	Jan Peters and Elmar Rueckert
Contributor(s):	Ryan Lober, Vincent Padois, Olivier Sigaud, Valerio Modugno, Gerhard Neumann, Elmar Rueckert, Giuseppe Oriolo, Jan Peters, Serena Ivaldi, Alex Paraschos
Reviewer(s):	
Approved by:	All Partners

Abstract	The scope of the current deliverable is to present the results on the learning of the task prioritization policies.
Keyword List:	contacts, inverse dynamics model learning, probabilistic movement representations, reinforcement learning

Document Revision History

Version	Date	Description	Author
v. 0.1	Feb. 07	Initial Draft	Elmar Rueckert
v. 0.2	Feb. 20	Draft	Elmar Rueckert
v. 1.0	Feb. 21	Final	Elmar Rueckert

Table of Contents

Chapter 1

Introduction

This deliverable presents results of task T4.4. In total three articles were presented at international robotics conferences (HUMANOIDS, ICRA and IROS) and one article is currently under review. Below, we discuss the achievements with respect to the task description from the Technical Annex.

1.1 Task description from the Technical Annex.

The core element of WP3 is the intelligent combination of prioritized tasks, which allows covering a large variety of possible scenarios while only requiring a small number of elements. Nevertheless, WP3s architecture requires a meaningful prioritization scheme that tells the systems which tasks to activate and how certain tasks can overrule each other. While it is possible to devise such prioritizations for complex tasks manually (see, e.g., Sentis et al., 2008), the automatic generation from data is much more desirable. Hence, in this task, we will investigate how a prioritization can be obtained from observing tasks, similar as in imitation learning, and how it can be self-improved. The relative importance of the tasks imposed by the prioritization can be changed during execution by the learned prioritization based on the current context. First, T4.4 will only help reproduce behavior from WP2 on the iCub but subsequently, it will allow for generalization to novel situations. Expected task outcomes are the following:

- Objective 1: A learned importance weighting for elementary tasks; weighting will allows appropriate combinations to generate solutions for new, more complex scenarios.
- Objective 2: A learned prioritization policy using both imitation and reinforcement learning (see tasks from WP2); demonstration and generalization to novel situations.

1.2 Contributions within the CoDyCo consortium.

We developed learning approaches that avoid task interferences prior to the task execution, that can be trained through reinforcement learning from a general task objective and from imitation learning in a probabilistic model.

- To objective 1: At UPMC, efficient whole-body control strategies have been developed that avoid interferences of multiple task objectives prior to a task execution. Two articles were published at international conferences and are summarized in Chapter 2. A detailed description can be found in *Deliverable D3.3*.
- To objective 2: In a collaboration, INRIA and TUDA studied the learning of task priority profiles for whole-body control. We optimized the parameters of priority profiles with respect to a general task objective through reinforcement learning [3]. This work is presented in Chapter 3.
  In another study, TUDA investigated how task such priority profiles can be obtained from imitation learning. A research article of this study is currently under review [4]. In Chapter 4, we present a draft of this contribution.

Chapter 2

Multiple Task Learning for Whole-Body Reactive Control (UPMC)

In the following two paragraphs, we summarize the work of Ryan Lober, Vincent Padois and Olivier Sigaud on learning the task prioritization of multiple tasks. For a detailed report on these two articles we refer to *Deliverable D3.3*.

Multiple Task Optimization using Dynamical Movement Primitives for Whole-Body Reactive Control. Whole-body controllers provide the tools to execute multiple simultaneous tasks on humanoid robots, but given the robots internal and external constraints, interferences are often generated which impede task completion. Priorities can be assigned to each task to manage these interferences, unfortunately, this is often done at the detriment of one or more tasks. In this paper we present a novel framework for defining and optimizing multiple tasks in order to resolve potential interferences prior to task execution and remove the need for prioritization. Our framework parameterizes tasks with Dynamical Movement Primitives, simulates and evaluates their execution, and optimizes their parameters based on a general compatibility principle, which is independent of the robots topology, tasks or environment. Two test cases on a simulation of a humanoid robot are used to demonstrate the successful optimization of initially interfering tasks using this framework.

This work was presented at the international conference on humanoid robots [1].

Variance Modulated Task Prioritization in Whole-Body Control. Whole-Body Control methods offer the potential to execute several tasks on highly redundant robots, such as humanoids. Unfortunately, task combinations often result in incompatibilities which generate undesirable behaviors. Prioritization techniques can prevent tasks from perturbing one another but often to the detriment of the lower precedence tasks. For many tasks, static prioritization is not necessary or even appropriate because tasks can often be achieved in variable ways, as in reaching. In this paper, we show that such task variability can be used to modulate task priorities during execution, to temporarily deviate certain tasks as needed, in the presence of incompatibilities. We first present a method for mapping from task variance to task priority and then provide an approach for computing task variance. Through three common conflict scenarios, we demonstrate that mapping from task variance to priorities reactively solves a number of task incompatibilities.

This work was presented at the international conference on intelligent robots and systems [2].

Chapter 3

Learning soft task priorities for control of redundant robots (INRIA & TUDA)

Learning soft task priorities for control of redundant robots

Valerio Modugno[1,4], Gerard Neumann[2], Elmar Rueckert[2], Giuseppe Oriolo[1], Jan Peters[2,3], Serena Ivaldi[2,4]

***Abstract*— One of the key problems in planning and control of redundant robots is the fast generation of controls when multiple tasks and constraints need to be satisfied. In the literature, this problem is classically solved by multi-task prioritized approaches, where the priority of each task is determined by a weight function, describing the task strict/soft priority. In this paper, we propose to leverage machine learning techniques to learn the temporal profiles of the task priorities, represented as parametrized weight functions: we automatically determine their parameters through a stochastic optimization procedure. We show the effectiveness of the proposed method on a simulated 7 DOF Kuka LWR and both a simulated and a real Kinova Jaco arm. We compare the performance of our approach to a state-of-the-art method based on soft task prioritization, where the task weights are typically hand-tuned.**

I. Introduction

Exploiting the redundancy in robotic systems to simultaneously fulfil a set of tasks is a classical problem for complex manipulators and humanoid robots [1], [2]. Several controllers have been proposed in the literature, where the tasks combination is determined by the relative importance of the tasks, expressed by the task priorities. There are two main approaches for prioritized multi-task controllers. The first is based on *strict task priorities*, where a hierarchical ordering of the tasks is defined: critical tasks (or tasks that are considered as more important) are fulfilled with higher priorities, and the low-priority tasks are solved in the null-space of the higher priority tasks [3], [4]. The second is based on *soft task priorities*, where the solution is typically given by a combination of weighted tasks [5]. The importance or "soft priority" of each individual task is represented by a scalar weight function, which evolves in time depending on the sequencing of the robot actions. By tuning the time-dependent vector of scalar weights, the global robot motion can be optimized. In simulation studies, it was shown that adapting these weights may result in a seamless transition between tasks (*i.e.*, reaching for an object, staying close to a resting posture and avoiding an obstacle), as well as in continuous task sequencing [6].

When complex robots, such as humanoids, need to perform manipulations while fulfilling many tasks and constraints (*e.g.*, tracking a desired trajectory, avoiding obstacles, controlling the interaction forces), the strict task priorities approaches typically require a priori a definition of the task hierarchy. For instance, Sentis and Khatib [7] defined three levels of hard priorities *i.e.*, constraints of utter importance (such as contacts, near-body objects, joint-limits and self-collisions), operational tasks demands (*i.e.*, manipulation and locomotion) and adaptable postures (*i.e.*, the residual motion). However, in many contexts, it is difficult to organize the tasks in a stack and pre-define their relative importance in forms of priorities. When priorities are strict, a higher-priority task can completely block lower-priority tasks, which can result in movements that are not satisfactory for the robot mission (*i.e.*, its "global" task). Another issue concerns the occurrence of discontinuities in the control law due to sudden changes in the prioritization [8].

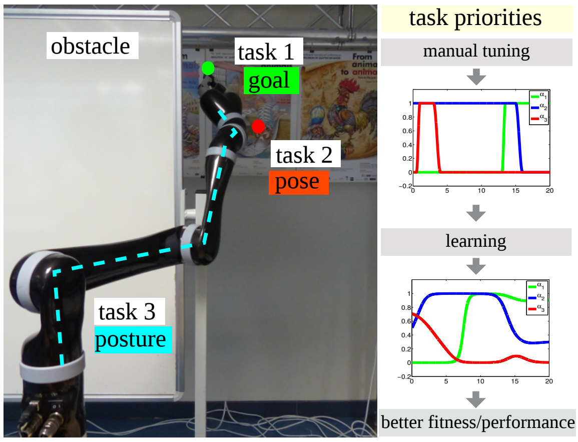


Fig. 1. The Jaco arm must reach a goal behind a wall (obstacle) while fulfilling a pose task on joint 4 and a full posture task. The initial sequencing of task priorities is not efficient. Our method allows the automatic learning of the temporal profiles of the task priorities from scratch.

Soft task priorities provide an appealing alternative solution. However, the simultaneous execution of different elementary tasks with variable soft priorities can lead to incompatibilities that might generate undesired movements or prevent the execution of some tasks. These issues are well explained in [9], where the authors modulate the task weights based on the movement variance to handle incompatibilities during online execution. Finally, when the number of tasks increases, for example in whole-body control of humanoid robots, and some tasks related to safety (*e.g.*, balance) are given high priority, it is generally difficult to define suitable task activations. In this case the priorities and their transitions are manually tuned by expert users [10] or defined before-hand [11]. Among the methods based on

*This paper was supported by the FP7 EU projects CoDyCo (No. 600716 ICT 2011.2.1 Cognitive Systems and Robotics).

[1] Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Universita di Roma, via Ariosto 25, 00185 Roma, Italy. modugno@diag.uniroma1.it

[2] Intelligent Autonomous Systems Lab., TU Darmstadt, Germany.

[3] Max Planck Institute for Intelligent Systems.

[4] Inria, Villers-lès-Nancy, F-54600, France; CNRS, Loria, UMR n.7503 and Université de Lorraine, Loria, UMR n.7503, Vandoeuvre-lès-Nancy, F-54500, France. serena.ivaldi@inria.fr

soft priorities, recently Liu et al. [6] proposed a generalized projector (GHC) that handles strict and non-strict priorities with smooth transitions when tasks priorities are swapped. Despite the elegant framework, their controller needs again a lot of manual tuning. The evolution of the tasks priorities in time, the timing and the tasks transitions need to be designed by hand. While this approach could still be easy for few tasks and simple robotic arms, it quickly becomes unfeasible for complex robots such as humanoids performing whole-body movements that usually require a dozen of tasks and constraints (*e.g.*, control balance, posture, end-effectors, stabilize head gaze, prevent slipping, control the interaction forces *etc.*). With the increasing abilities of humanoid robots, the number of tasks increases, together with their weights or priorities: their manual tuning through a sequence of complex manipulations becomes a major bottleneck.

In this paper, we propose a framework that addresses the issue of automatically optimising the task priorities by means of a learning algorithm. The proposed concept of learning the soft priorities can be applied to existing multi-task frameworks, such as the GHC [10]. However, we use here a simpler controller based on a regularized version of the Unified Framework for Robot Control (UF) [13] proposed by Peters *et al*. In our framework, the task weight functions are parametrized functional approximators that can be automatically learned by state-of-the-art stochastic optimization algorithms. The *temporal* profiles of the task weights can be learned by optimizing a given fitness function, used to evaluate the performance of candidate task priorities. In contrast to many cost functions used in whole-body optimisation frameworks, here we do not require the fitness to be a linear or quadratic function.

We show the effectiveness of our approach on both a simulated and a real 6 degrees of freedom (DOF) Kinova Jaco arm, on a goal reaching problem with several elementary tasks. Furthermore, we compare the performance of our controller with the state-of-the-art method GHC proposed by Liu *et al.* [10] on a simulated 7 DOF Kuka LWR arm.

The paper is organized as follows. Section II presents the proposed approach, describing the structure of the controllers, the task weight functions and the learning procedure. We present the experimental results in Section III, draw conclusions and discuss future work in Section IV.

II. Methods

Let us consider a “global task” or a “mission” for a redundant robot: for example, *to reach a goal point behind a wall while avoiding an obstacle*. The overall movement can be entirely planned by exploring all the possible joint configurations, or it can be generated by a combination of a set of controllers solving simpler elementary tasks (for example: control the end-effector, control the pose of a particular link, *etc.*). We assume that the set of elementary tasks is known, and that each task can be executed by a given torque controller. The global movement can be evaluated by a fitness function ϕ that can be used as a measure of the ability of the robot to fulfil its mission. Our method aims at automatically learning the task priorities (or task weight functions) to maximize the robot performance.

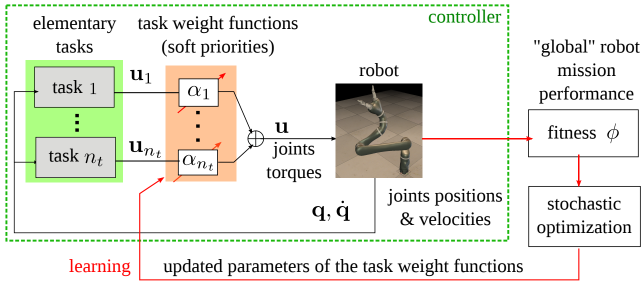


Fig. 2. Overview of the proposed method. The controller consists of a weighted combination of elementary tasks, where the weight functions represent the soft task priorities. An outer learning loop enables the optimization of the task weights parameters.

An overview of the proposed approach is illustrated in Fig. 2. In Section II-A we describe the controller $\mathbf{u}_i$ for each elementary task: a regularized version of the Unified Framework [13]. In Section II-B we describe the multi-task controller with learned task priorities. In Section II-C we describe the parametrized task weight functions α_i used by our multi-task controller, and discuss the parameters optimization. As a learning algorithm, we propose the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [14], a derivative-free stochastic optimization algorithm, in view of its good exploration properties and ease of use.

A. Controller for a single elementary task

We hereby describe the torque controller for the i-th elementary task. To simplify the controller design, we decided to adopt the Unified Framework (UF) [13]. We consider the well-known rigid-body dynamics of a robot with n DOF, *i.e.*,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}), \tag{1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}} \in \mathbb{R}^n$ are, respectively, the joints positions, velocities and accelerations, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the generalized inertia matrix, $\mathbf{f}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^n$ accounts for Coriolis, centrifugal and gravitational forces and $\mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^n$ is the vector of the commanded torques of the i-th task. Using the same notation as in [13], we describe the task as a constraint, given by $\mathbf{h}_i(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{0}$, where $\mathbf{h}_i \in \mathbb{R}^m$ is at least a twice differentiable function, where m is the task dimension. By differentiating the constraint with respect to time, we obtain:

$$\mathbf{A}_i(\mathbf{q},\dot{\mathbf{q}},t)\ddot{\mathbf{q}} = \mathbf{b}_i(\mathbf{q},\dot{\mathbf{q}},t), \tag{2}$$

where $\mathbf{A}_i(\mathbf{q},\dot{\mathbf{q}},t)$ is a known $m \times n$ matrix and $\mathbf{b}_i(\mathbf{q},\dot{\mathbf{q}},t)$ is a $m \times 1$ vector. For example, given a simple tracking control task with $\mathbf{h}_i(\mathbf{q},\dot{\mathbf{q}},t) = \mathbf{q} - \mathbf{q}^{des}$, where $\mathbf{q}^{des}$ corresponds to a desired trajectory. By computing the second order derivative in t we obtain $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}^{des}$, where $\mathbf{b} = \ddot{\mathbf{q}}^{des}$ and $\mathbf{A}_i = \mathbf{I}$ (with $\mathbf{I}$ the identity matrix). Applying Gauss's principle, it is possible to derive a controller that fulfils the constraints by minimizing the cost function $\zeta_i(t) = \mathbf{u}_i^\top \mathbf{N}_i(t)\mathbf{u}_i$, where $\mathbf{N}_i(t)$ is a positive semidefinite matrix. The optimization problem is defined by

$$\mathbf{u}_i^* = \arg\min_{\mathbf{u}_i} \zeta_i(t) = \arg\min_{\mathbf{u}_i} \left[\mathbf{u}_i^\top \mathbf{N}_i(t)\mathbf{u}_i\right], \tag{3}$$

subject to Eq. 1 and 2. The solution to this optimization problem is given by

$$\mathbf{u}_i = \mathbf{N}_i^{-\frac{1}{2}}(\mathbf{A}_i\mathbf{M}^{-1}\mathbf{N}_i^{-\frac{1}{2}})^{\#}(\mathbf{b}_i + \mathbf{A}_i\mathbf{M}^{-1}\mathbf{f}), \tag{4}$$

where $(\cdot)^{\#}$ is the Moore-Penrose pseudoinverse and the upper script in $\mathbf{N}_i^{-\frac{1}{2}}$ denotes the inverse of the matrix square root. Controllers derived from UF are sensitive to kinematic singularities, due to the matrix inversion [15]. To overcome this problem, we reformulate the UF controller in a *regularized* fashion, as classically done at the kinematic level, for instance in [16]. The objective function of UF can be reformulated in such a way that the solutions of the optimization problem naturally exhibit a damped least squares structure (at the price of a loss of precision in the execution of the elementary task). Given the dynamical model of the robot (Eq. 1) and the constraint that describes the task (Eq. 2), we define the regularized optimal control problem:

$$\arg\min_{\mathbf{u}_i} \zeta_i(t) = \arg\min_{\mathbf{u}_i} \left[(\mathbf{A}_i\ddot{\mathbf{q}} - \mathbf{b}_i)^2 + \mathbf{u}_i^{\top} \frac{\mathbf{N}_i(t)}{\lambda_i} \mathbf{u}_i \right], \tag{5}$$

where λ_i is the regularizing factor with a l^2-weighted norm for the regularization term. In the simplest case, λ_i can be a manually-tuned constant value, or automatically determined by more sophisticated methods, as done in [17], based on the smallest singular value of the matrix to invert. To derive the closed form solution of the optimization problem, we substitute $\ddot{\mathbf{q}}$ in Eq. 5 with the expression obtained by solving the dynamical constraint for $\ddot{\mathbf{q}}$. The resulting closed form solution of the controller for a single elementary task is then:

$$\mathbf{u}_i = \mathbf{N}_i^{-1}\tilde{\mathbf{M}}_i^{\top}(\mathbf{I}\lambda_i^{-1} + \tilde{\mathbf{M}}_i\mathbf{N}_i^{-1}\tilde{\mathbf{M}}_i^{\top})^{-1}(\mathbf{b}_i + \tilde{\mathbf{M}}_i\mathbf{f}), \tag{6}$$

with $\tilde{\mathbf{M}}_i = \mathbf{A}_i\mathbf{M}^{-1}$.

B. Controller for multiple elementary tasks with soft task priorities

With reference to the scheme of Fig. 2, we consider a number n_t of elementary tasks, that can be combined by the robot to accomplish a given "global" mission. The solution of the i-th task is provided by the torque controller $\mathbf{u}_i$ described in the previous section. Each task is modulated by a task priority or task weight function $\alpha_i(t)$. The ensemble $\{\alpha_i(t)\}_{i=1,\dots,n_t}$ constitutes the *activation policy* that determines the overall robot movement. The robot controller is therefore given by

$$\mathbf{u}(\mathbf{q},\dot{\mathbf{q}},t) = \sum_{i=1}^{n_t} \alpha_i(t)\,\mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}) \ , \tag{7}$$

where t is the time, and $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the robot joint positions and velocities. The task priorities $\alpha_i(t)$ are scalar functions and their time profile can be optimized. We automatically tune the task priorities with a learning algorithm. We seek the best task weight functions that maximize a defined performance measure, or fitness, evaluating the execution of the global task. As finding the optimal functions $\alpha_i^*(t)$ is an intractable problem, we turn the functional optimization problem into a numerical optimization problem by representing the task priorities with parametrized functional approximators, $\alpha_i(t) \rightarrow \hat{\alpha}_i(\boldsymbol{\pi}_i,t)$, where $\boldsymbol{\pi}_i$ is the set of parameters that shape the temporal profile of the i-th task weight function. The controller then becomes:

$$\mathbf{u}(\mathbf{q},\dot{\mathbf{q}},t) = \sum_{i=1}^{n_t} \hat{\alpha}_i(\boldsymbol{\pi}_i,t)\,\mathbf{u}_i(\mathbf{q},\dot{\mathbf{q}}) \tag{8}$$

Finding the optimal task priorities consists therefore in finding the optimal parameters $\boldsymbol{\pi}_i^*$, which can be done applying a learning method to maximize the fitness ϕ.

C. Learning the task priorities

We model the task priorities by a weighted sum of normalized Radial Basis Functions (RBF):

$$\hat{\alpha}_i(\boldsymbol{\pi}_i,t) = S\left(\frac{\sum_{k=1}^{n_r} \pi_{ik}\psi_k(\mu_k,\sigma_k,t)}{\sum_{k=1}^{n_r} \psi_k(\mu_k,\sigma_k,t)}\right), \tag{9}$$

where $\psi_k(\mu_k,\sigma_k,t) = \exp\left(-1/2[(t-\mu_k)/\sigma_k]^2\right)$, with (μ_k,σ_k) being mean and variance of the basis functions, n_r is the number of RBFs and $\boldsymbol{\pi}_i = (\pi_{i1},\dots,\pi_{in_r})$ is the set of parameters of each task priority. $S(\cdot)$ is a sigmoid function that squashes the output to the range $[0,1]$. When the task priority is 1, the task is fully activated; when its value is 0, the task is not active.

In our method, learning the task priorities is implemented by learning the free parameters $\boldsymbol{\pi}_i$ of the weight functions (Eq. 9). We optimize the parameters with respect to a known fitness function $\phi = \phi(\mathbf{q}_{t=1,\dots,T},\mathbf{u}_{t=1,\dots,T},t)$, given T time steps. ϕ describes the performance of the controller in fulfilling its global mission. The fitness function could be a simple measure of success in case of goal reaching, the time duration of a movement, the energy consumption *etc.* More criteria for optimizing robot motions in optimal control frameworks are reported in [18]. If the fitness function ϕ is differentiable with respect to the controls and the parameters (which requires the function approximators to be differentiable as well with respect to the controls [17]), then gradient methods can be used. If the fitness is not differentiable with respect to the parameters, then a derivative-free method can be used. Thus, derivative-free methods are appealing, since they do not constrain the design of the fitness function. Furthermore, recent results showed that it is possible to achieve very fast performances in trial-and-error learning with derivative-free methods [19].

As optimization algorithm, we use CMA-ES [14], which is a stochastic derivative-free optimization strategy that is suitable for non-linear and non-convex objective functions. This method belongs to the class of evolutionary algorithms. At each iteration of the algorithm, new candidate solutions are generated from a population of candidates through stochastic sampling. The fitness function is then used to evaluate the candidates. In our case, each candidate is a possible set of parameters for the task priorities $\mathbf{x} = \{\boldsymbol{\pi}_1,\dots,\boldsymbol{\pi}_{n_t}\}$ (Eq. 9). At each iteration of the algorithm (called *generation*), a new population of candidates is generated by sampling a multivariate normal distribution $\mathcal{N}(\mathbf{m},\boldsymbol{\Omega})$, where $\mathbf{m}$ and

$\mathbf{\Omega}$ represent respectively mean and covariance of the distribution. A fitness value is computed for each candidate in the current population and, according to the fitness, only the most promising candidates are kept to update the search distribution. Given the n_c candidates $\{\mathbf{x}_1,\ldots,\mathbf{x}_{n_c}\}$, the algorithm selects the $n_b < n_c$ best ones according to their ordered fitness values $\{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_{n_b}\}$. It uses the selected candidates to compute the mean of the sampling distribution at the next iteration: $\mathbf{m}^{(new)} = \sum_{i=1}^{n_b} \omega_i \hat{\mathbf{x}}_i$, with $\sum_{i=1}^{n_b} \omega_i = 1$. Then the covariance matrix is updated as:

$\mathbf{\Omega}^{(new)} = (1-c_1-c_2)\mathbf{\Omega} + c_1 p_\Omega p_\Omega^\top + c_2 \sum_{i=1}^{n_b} \omega_i \mathbf{y}_i (\mathbf{y}_i)^\top$ with $\mathbf{y}_i = (\hat{\mathbf{x}}_i - \mathbf{m})$, c_1 and c_2 two predefined parameter (see [14] for more details). The symbol p_Ω is a term measuring the correlation among successive generations. The covariance is related to the *exploration rate* of the algorithm, a scalar value between [0,1] and the only parameter of the algorithm that needs to be tuned. This version of CMA-ES does not support constrained optimization, which means that the optimized solutions that are not physically feasible on the real robot must be dropped and the learning algorithm restarted. In the follow-up of this work, we will use a version that supports constraints [20].

III. Experiments

In this section we discuss our experiments on learning the task priorities. We start showing on a simulated Kinova Jaco arm that the our learning method improves the performance of the movement in terms of fitness values, over existing task priorities that have been manually tuned. We also show that the optimized trajectories are robust with respect to the initialization of the learning process. We compare on a real Jaco arm some typical learned policies with the manually tuned one, showing that our method improves the real robot motion. Finally, we compare on a simulated Kuka LWR the performance of our method with the state-of-art GHC controller [6] . We show that our method is not only better in terms of performance, but also computationally 10 times faster.

A. *Learning the task priorities for the Kinova Jaco arm*

The setting for the first experiment is shown in Figure 1. The Kinova Jaco arm (6 DOF), starting from its zero configuration, must reach a desired position behind a wall with its end-effector. The goal position is difficult to reach, and the robot kinematics is such that it is not straightforward to manually design a trajectory that does not collide with the obstacle and brings the hand to the goal.

There are 3 given elementary tasks. The first is about reaching the Cartesian position $\mathbf{p}^*$ =[0, -0.63, 0.7] with the end-effector (*goal*). The second is about reaching the Cartesian pose [-0.31, -0.47, 0.58] with the 4th link. The third is about keeping the joint configuration [120, 116, 90, 0, 0, 0] (degrees).We design the following fitness function $\phi \in [-1,0]$:

$$\phi(\mathbf{q}_{1,\ldots,T},\mathbf{u}_{1,\ldots,T}) = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_i - \mathbf{p}^*\|}{\varepsilon_{\max}} + \frac{\sum_i^T \|\mathbf{u}_i\|_2^2}{u_{\max}}\right)$$

where T is the number of control steps (the task duration is 20 seconds, and we control at 10ms), $\mathbf{p}_i$ describes the end-effector position at time i and $\mathbf{p}^*$ is the goal position, $\|\cdot\|_2^2$ is the square of the ℓ^2 norm and $\varepsilon_{\max}^{-1}$ and $u_{\max}^{-1}$ are two scaling factors. The first term of ϕ penalizes the cumulated distance from the goal that enforces a minimum time transfer trajectory for the robot arm, while the second term penalizes the global control effort. To ensure that the generated controls are feasible for the real Jaco robot, we set the fitness to -1 whenever the generated policy violates one of the robot constraints: a collision with the environment, joints position ranges and maximum joint torques. This ad-hoc solution is also a consequence of the learning algorithm.

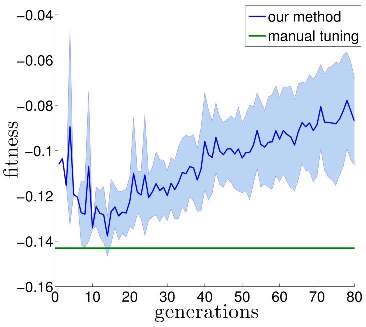


Fig. 3. Average fitness value for the task priorities learned with our method, for the 3-tasks experiment with the simulated Jaco arm. The horizontal line indicates the fitness of the manually tuned solution. The mean and standard deviation of the fitness for the learned policies is computed over 100 restarts of CMA-ES, each with 80 generations and random initialization of the parameters. We only retained the fitness for the experiments that provided solutions satisfying the real robot constraints.

Fig. 3 shows the average fitness value computed over the eleven optimized trajectories satisfying the constraints, found on 100 restarts of CMA-ES.

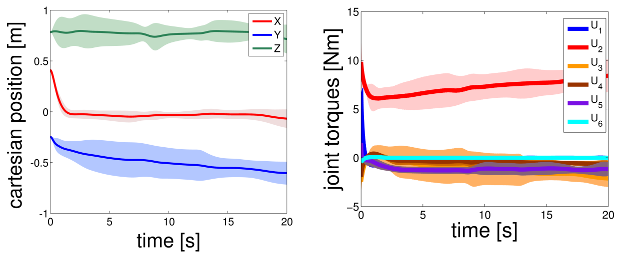
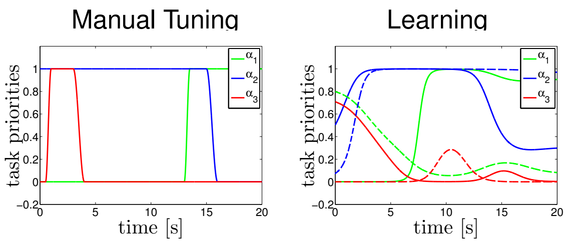


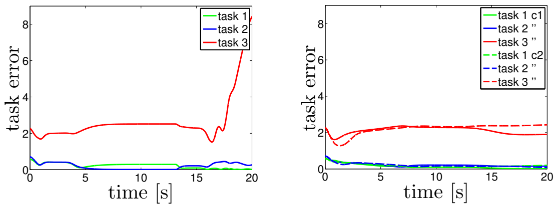
Fig. 4. Mean and variance of the Cartesian trajectory of the end-effector and the joints torques of the simulated Jaco arm, generated by learned task priorities over 100 trials of the 3-tasks experiment (see text in Section III-B). Even starting from random initialization of the task weight parameters, the learning process is eventually able to produce similar optimized motions of the robot that fulfil its 'global' task.

B. *Robustness of the learning process*

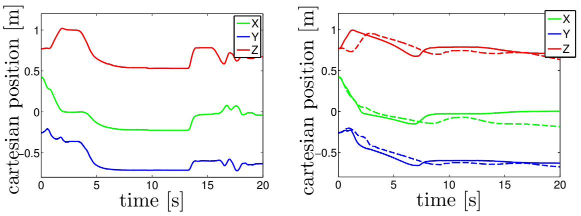
Different profiles of the task priorities can yield similar movements of the robot. It is however important to show that the learning process is able to optimize the task priorities in a robust way, that is providing similar optimal solutions. We therefore execute N=100 replicates of the experiment in Section III-A, with a simulated Jaco arm and three tasks. In each experiment, CMA-ES runs for 100 generations with an exploration rate of 0.5. The parameters are randomly



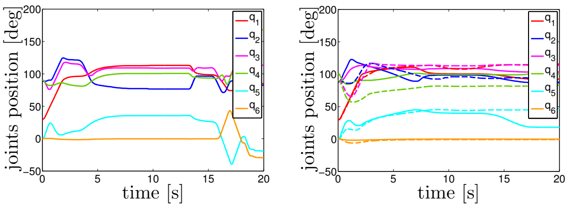
(a) The task priorities evolving in time.



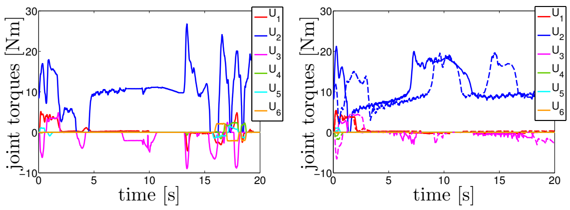
(b) The task errors.



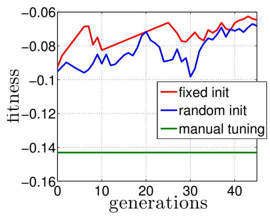
(c) The end-effector trajectory in the Cartesian space.



(d) The joints trajectories.



(e) The measured joints torques (estimated by the motor currents).



(f) The fitness values.

Fig. 5. Comparison between a manually tuned (left side) and two typical learned (right side) policies for the 3-tasks experiment performed with the real Kinova Jaco arm. On the right, a solid line corresponds to a policy optimized starting from a fixed/known initial value of the priorities (*fixed init*), in this case the priorities found by manual tuning; the dashed line corresponds to a policy optimized starting from random values (*random init*). The final fitness values are: -0.1431 (manual tuning), -0.0585 (fixed init) and -0.0644 (random init).

initialized. We compute the average and standard deviation of the solutions that satisfy the robot and task constraints. Figure 4 shows the average end-effector trajectory in the Cartesian space and the corresponding joint torques. Despite the redundancy of the robot and the one of the task priorities, the final robot movements are smooth and quite consistent with each other. Their average fitness is -0.0874 ± 0.0213. Overall, this result indicates that learning the soft task priorities starting from scratch (*i.e.*, where an initial guess for the activation of the task priorities in time is not available) is a viable and robust option for generating the motion of redundant robots.

C. Experiment on the real Kinova Jaco arm

We compare in Fig. 5 the effect of three different task prioritizations on the real Jaco arm. In the left column, we show the robot movement generated by task priorities that were manually tuned by an expert user of the Kinova arm; on the right column, we show two typical robot movements generated by learned task priorities, which were optimized with CMA-ES starting from a known initial value (the manually tuned task weight functions) and a random value. We set the exploration rate in CMA-ES to 0.5 and perform 40 generations. Learning the priorities has a beneficial effect on the smoothness of the trajectories, which becomes evident when comparing the plots of the end-effector (Fig. 5c), joints positions (Fig. 5d) and torques (Fig. 5e) and the task errors (Fig. 5b). We evaluate the fitness using the commanded joint torques $\mathbf{u}_i$ and the kinematics and dynamics model of the Jaco arm to compute $\mathbf{p}_i$. The fitness value for the manually tuned task priorities is -0.1431. The fitness values for the two optimized solutions are better: -0.0585 and -0.0644 initializing the parameters with fixed and random values respectively. Overall, this experiment illustrates that learning the task priorities improves the real robot motion with respect to an existing manually tuned solution.

D. Comparison with the state-of-the-art GHC

In this experiment we compare the performance of the task priorities learning applied to our method and to the state-of-the-art multi-task controller GHC [6].

In the GHC, each task is associated to a null space projector of the extended Jacobian that contains the analytical description of all the task objectives. Soft task prioritization is achieved because the null space projector depends on a set of manually designed weight functions ranging from 0 to 1, that control if each task is fully or partially projected in the null spaces of the other tasks with higher priority. The controller is the solution to a quadratic optimal control problem subject to the robot and task constraints – see [6]. The soft task priorities are introduced as a further constraints, formulated by $\ddot{\mathbf{q}} = \sum_{i=1}^{n_t} \mathbf{P}_i(\mathbf{\Lambda}_i)\ddot{\mathbf{q}}_i^{'}$, where $\mathbf{P}_i(\cdot)$ is the null space projector associated to the i-th task, $\mathbf{\Lambda}_i$ is a matrix that depends on the task priorities, $\ddot{\mathbf{q}}_i^{'}$ are intermediate joint accelerations associated to each task and $\ddot{\mathbf{q}}$ are the joints accelerations. To enable the comparison with our method,

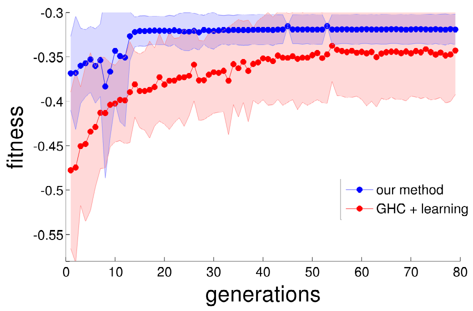


Fig. 6. Comparison between our method and the GHC modified to learn the task priorities with CMA-ES. The plot shows the mean and the standard deviation of the fitness in $R = 20$ trials of the experiment with the simulated KUKA LWR arm (see Section III-D). For both controllers, the learning is initialized with random parameters. Our method shows a faster convergence and better optimization of the fitness. The average fitness is is -0.0373 ± 0.0320 for our method and -0.0735 ± 0.0946 for the GHC+learning. The two distributions are statistically different ($p < 0.01$ with the K-S test).

we parametrized the task priority matrix $\mathbf{\Lambda}_i$ for each task i in the same way as described in Section II-B.

We compare the two methods on a reaching task with a simulated 7 DOF Kuka LWR, which was originally used in [6]. In this scenario, the robot must reach a goal point beneath a rectangular surface parallel to the ground ($z = 0.25m$), without collision. The are 2 elementary tasks. The first is about reaching the Cartesian position $\mathbf{p}^*$ =[0.6, 0, 0.15] with the end-effector (*goal*). The second is about keeping the joint configuration [0, 90, 0, -90, 0, 90, 0] (degrees). We design the following fitness function $\phi \in [-1, 0]$:

$$\phi(\mathbf{q}_{1,\ldots,T}, \mathbf{u}_{1,\ldots,T}) = -\frac{1}{2}\left(\frac{\sum_i^T \|\mathbf{p}_i - \mathbf{p}^*\|}{\varepsilon_{\max}} + \frac{\max(\|\mathbf{u}_{i=1,\ldots,T}\|_\infty)}{u_{\max}}\right)$$

where $\|\cdot\|_\infty$ is the infinity norm. We set the fitness to -1 in case of collision. We run 20 experiments from random initial parameters for both methods, with an exploration rate of 0.1 for CMA-ES and 80 generations.

Our method generated solutions that satisfy the collision constraint in 90% of the cases, while the GHC succeeded only in 75%. Figure 6 shows the mean and standard deviation of the fitness: our method is faster in convergence and improves the final optimized fitness. The average fitness at the end of the learning process is -0.0373 ± 0.0320 for our method and -0.0735 ± 0.0946 for the GHC+learning. The two distributions of the fitness are statistically different ($p = 0.0073 < 0.01$, obtained with the two-sample Kolmogorov-Smirnov test). Our method is also 10 times faster in terms of computational time: on a standard i7 machine, the average time for the optimization process to find a solution with 80 generations (average over 20 trials) is $3.7 \times 10^3 \pm 2.4 \times 10^2$ seconds for our method and $3.9 \times 10^4 \pm 2.2 \times 10^3$ seconds for the GHC+learning approach.

IV. Conclusion and Future Work

In this paper we address an important issue for prioritized multi-task controllers, that is the automatic and optimal generation of task priorities through parametrized weight functions. As a first step towards an automatically tuned controller for redundant robots, we propose a novel framework with a multi-task controller where the task priorities can be learned via stochastic optimization. We show the effectiveness of our approach by comparing to GHC [10], a state-of-the-art multi-task prioritized controller. We present several results performed on a simulated 7 DOF Kuka LWR arm and both a simulated and a real 6 DOF Kinova Jaco arm. Ongoing work is focused on improving the current framework from different points of view: addressing generalization, using constraints inside the optimization, and scaling up the method to handle robots with several DOF, e.g., humanoid robots.

References

[1] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *IJRR*, vol. 6, no. 2, pp. 3–15, 1987.
[2] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Int. Conf. Advanced Robotics*, 1991, pp. 1211–1216.
[3] L. Saab, O. Ramos, F. Keith, N. Mansard, P. Soueres, and J.-Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Trans. on Robotics*, vol. 29, pp. 346–362, Jan 2013.
[4] A. Del Prete, F. Nori, G. Metta, and L. Natale, "Prioritized motion-force control of constrained fully-actuated robots: Task space inverse dynamics," *Robotics and Autonomous Systems*, vol. 63, pp. 150–157, Jan 2015.
[5] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *ICRA*, 2011, pp. 1283–1290.
[6] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, pp. 1–15, 2015.
[7] L. Sentis and O. Khatib, "Synthesis of whole body behaviours through hierarchical control of behavioral primitives," *Int. Journal of Humanoid Robotics*, pp. 505–518, 2005.
[8] C. Ott, A. Dietrich, and A. Albu-Schffer, "Prioritized multi-task compliance control of redundant manipulators," *Automatica*, vol. 53, pp. 416 – 423, 2015.
[9] R. Lober, V. Padois, and O. Sigaud, "Variance modulated task prioritization in whole-body control," in *IROS*, 2015, pp. 1–6.
[10] M. Liu, S. Hak, and V. Padois, "Generalized projector for task priority transitions during hierarchical control," in *ICRA*, 2015, pp. 768–773.
[11] S.-I. An and D. Lee, "Prioritized inverse kinematics with multiple task definitions," *ICRA*, pp. 1423–1430, 2015.
[12] J. Kober, D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *IJRR*, vol. 11, pp. 1238–1274, 2013.
[13] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal, "A unifying framework for robot control with redundant dofs," *Autonomous Robots*, vol. 24, pp. 1–12, Jan 2008.
[14] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies." *Evolutionary Computation*, vol. 9, pp. 159–195, Jan 2001.
[15] S. Chiaverini, B. Siciliano, and O. Egeland, "Redundancy resolution for the human-arm-like manipulator," *Robotics and Autonomous Systems*, vol. 8(3), pp. 239–250, Jan 1991.
[16] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *J. Dyn. Sys., Meas., Control*, vol. 108 (3), pp. 163–171, 1986.
[17] S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta, and G. Sandini, "Approximate optimal control for reaching and trajectory planning in a humanoid robot," in *IROS*, 2010, pp. 1290–1296.
[18] S. Ivaldi, O. Sigaud, B. Berret, and F. Nori, "From humans to humanoids: the optimal control framework," *Paladyn Journal of Behavioral Robotics*, vol. 3, no. 2, pp. 75–91, 2012.
[19] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.
[20] D. V. Arnold and N. Hansen, "A (1+ 1)-cma-es for constrained optimisation," in *GECCO*, 2012, pp. 297–304.

Chapter 4

Probabilistic Prioritization of Movement Primitives (TUDA)

Probabilistic Prioritization of Movement Primitives

Alexandros Paraschos[1], Jan Peters[1,2] and Gerhard Neumann[1]

Abstract— **Movement prioritization is a common approach to combine controllers of different tasks for redundant robots. Each task is assigned a priority, where either strict or "soft" priorities can be used. While movement prioritization is an important concept in the control of whole body movements, it has been less considered in learning-based approaches, where prioritization allows us to learn different tasks for different end-effectors, and subsequently reproduce an arbitrary, unseen combination of these tasks. This paper combines Bayesian task prioritization, a "soft" prioritization technique, with probabilistic movement primitives to prioritize full motion sequences. Probabilistic movement primitives can encode distributions of movements over full motion sequences and provide control laws to exactly follow these distributions. The probabilistic formulation allows for a natural application of Bayesian task prioritization. We demonstrate how the "soft" priorities can be obtained from imitation learning and that our prioritized learning architecture can reproduce unseen task-combinations. Moreover, we require less data to learn a combination of tasks than the traditional approach that directly models each task in joint space. We evaluate our approach on reaching movements under constraints with a redundant bi-manual planar robot and the humanoid robot iCub.**

I. Introduction

Complex robots with redundant degrees of freedom have increased manipulation capabilities and, can in principle perform multiple tasks at the same time. For example, possible task combinations are reaching an object with a humanoid robot while balancing or reaching an object with a robotic arm while the "elbow" avoids an obstacle.

Performing multiple tasks simultaneously is not trivial, as it often requires to simultaneously control the same joints with different control laws, that leads to conflicting control signals. Many control schemes that can combine these signals were developed. We focus on approaches that resolve the control combination problem by prioritizing the tasks. Task prioritization can be either strict, where a lower priority task is not allowed to interfere with higher priority tasks, or "soft", where the aforementioned assumption can be violated. Movement prioritization is an established concept for controlling whole body movements, however, prioritized task combination is also a powerful concept for learning-based approaches, where such concepts have not yet been explored so far. For example, prioritization allows us to learn different tasks for different end-effectors and subsequently reproduce an arbitrary, unseen combination of these tasks.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreements #600716 (CoDyCo) and #270327 (CompLACS)

[1]Intelligent Autonomous Systems, TU Darmstadt, 64289 Darmstadt, Germany `{paraschos,neumann}@ias.tu-darmstadt.de`

[2] Robot Learning Group, Max Planck Institute for Intelligent Systems, Germany `mail@jan-peters.net`

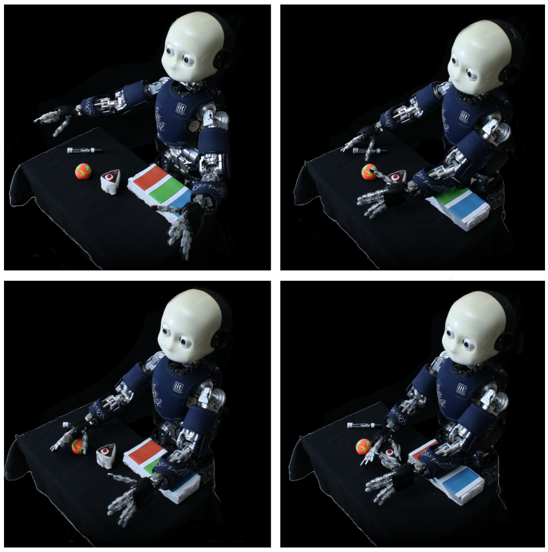


Fig. 1. The iCub robot performing a bi-manual reaching task. With the left end-effecotr, the robot initiates a supportive contact with the environment, while it performs a reaching task with the right end-effector. We illustrated our setup if the first picture. The robot stands on the floor and had the choice of three supporting contact locations, shown in blue, green, and red. With the right end-effector the robot can reach for grasping three different objects, a pen, a ball, and a piece of cake. In the remaining pictures we present our results for learning and generalizing to new task combinations of reaching and contact support locations.

In this paper, we propose a new imitation learning architecture that learns a prioritized skill representation. We combine Bayesian task prioritization [1], a "soft" prioritization method, with Probabilistic Movement Primitives (ProMPs) [2], [3] to prioritize motion sequences that are learned from demonstrations. Bayesian task prioritization has been introduced in [1] but gained little attention in that paper and follow up work. We provide a more general derivation for torque control and show that existing prioritization techniques are a special case of the Bayesian approach.

Our prioritization is data-driven, i.e. it employs demonstrations to the robot to extract the relative prioritization from imitation data. The demonstrations can be acquired by several imitation learning techniques, including kinesthetic teach-in and tele-operation. We use multiple demonstrations for every task to accurately extract the variance of its movement, where different tasks can be learned for different end-effectors. In contrast to other approaches, we present a closed form solution for setting each task's "soft" priority from the variance of the task in its operational space. Furthermore, we compute the output control, still in closed form. We represent the variance of a task at each state using ProMPs. ProMPs encode a trajectory distribution and, thus, represent the time-varying variance of each task. The primitives can be trained using imitation learning, and, generate probabilistic controllers that follow exactly the encoded task distribution. The variances of the probabilistic controllers can be used naturally for Bayesian task prioritization.

We can learn multiple primitives for a single end-effector, where each primitives solves a specific task with the corresponding end-effector. The primitives of different end-effectors can now be seamlessly combined in order to achieve a new, unseen combination of tasks of the end-effectors. Another advantage of ProMPs is that we can adapt the distribution by conditioning on reaching different via-points. Using the prioritization of ProMPs, conditioning can now be done also for task space variables instead of simply in joint space. We demonstrate that our data-driven prioritization approach can be used for conditioning in task-space as well as the improved multi-task learning capabilities or our approach in simulation and on a reaching and stabilizing task with the iCub.

II. Related work

A common resolution for combining different control signals is to prioritize the tasks, under the assumption that this prioritization is not allowed to be violated. We refer to such schemes as a strict prioritization schemes. In these schemes, a higher priority task does not get disturbed by the control signals of the lower priority ones [4], [5], [6], [7], [8], [9], [10], [11], [12]. A lower priority task is always projected in the null-space of the high priority task. Although these approaches provide guarantees on the system performance, they are often over-constraining and therefore limit its usefulness in real-word applications. Moreover, strict prioritization approaches might get numerically instable when the robot enters a singular kinematic configuration. Proposed solutions to the numerical stability problem exist [13], [14], but also have a side-effect: they relax to some extent the assumption that a low priority task does not interfere with a higher priority task.

For some tasks, such a prioritization scheme is natural, for example, a humanoid robot should not tip over and, therefore, the balancing controller should always have the highest priority. However, defining a strict priority can be problematic in general. For example, for reaching an object with one hand of a humanoid robot, while simultaneously reaching for a different object with the other hand, it is not clear these tasks can be prioritized. Both tasks could have the same importance, i.e. priority, or, the importance of each tasks could vary in time and depending on, e.g., the desired execution accuracy at that time point. For such scenarios, the relative importance between the tasks is easier to set and requires less hand-tuning. These problems are partially addressed in [15], [16], [17], where a “soft” prioritization scheme was introduced. In our approach, we step further and propose to learn the relative priorities from data and, therefore, minimize the amount of parameters that require expert knowledge to be tuned.

“Soft” prioritization approaches do not assume a priori a hierarchy of tasks but they use the relative priorities between the tasks. In this scheme, every task contributes to the control signal. The degree of contribution depends on its relative priority. “Soft” prioritization approaches demonstrate promising results where strict approaches fail [15], [18], [19] to successfully perform multiple tasks due to the relaxation of the initial problem. Intuitively, “soft” prioritization schemes could be thought as violating the hierarchy of priorities. They are often formulated as multi-objective optimization problems [15], [18], [19]. Each task is formulated as a quadratic cost function and uses the relative priority as weight. The result of the optimization yields the controls that minimize the total cost and, therefore, allows lower priority tasks to perturb higher priority ones as long as the total cost is decreased.

Both strict and “soft” prioritization approaches often assume a static prioritization or weighting scheme, where the importance of each task remains constant during the execution of the movement [6], [17]. However, modulating the importance of the tasks during the movement can be beneficial. First, tasks that are no longer desired to be executed can be faded-out and new tasks can be smoothly introduced, without torque jumps. Salini et al. [15] proposed to dynamically adjust the priorities for achieving movement sequencing and tasks transitions. Second, and more importantly, the modulation of the priorities can be related to the desired accuracy of the task. During the time-steps with low task-priority, the robot can focus on executing other tasks. Therefore, setting the relative priorities can be a simpler problem then specifying the strict task hierarchy, as the expert has to specify only the time points that require higher accuracy. Lober et al. [19] demonstrated that this approach increases the flexibility of the system and decreases “lock-ups” where a more important movement prohibits the execution of less important tasks, while it requires less expert knowledge. Modugno et al. [20] proposed the use of an optimization algorithm to find suitable “soft” priorities that further decreases expert knowledge.

Movement primitives are a well established concept for imitation learning and generalization of movements in robotics [21], [22], [23], [2], however, no primitive representation has so far taken leverage from introducing task priorities. In this paper, we introduce for the first time task-priorities for movement primitives. We use the Probabilistic Movement Primitive approach as it can be naturally combined with Bayesian task prioritization in a single probabilistic framework. However, other stochastic movement representations could also be used instead [22], [24].

III. Probabilistic Prioritization of Stochastic Controllers

In this section, we develop a generic probabilistic framework for simultaneously combining multiple tasks. We assume that each tasks has a different degree of accuracy and that the accuracy changes over time. We associate the task accuracy with its importance for the task combination.

First, we show how the time-varying task accuracy can be encoded in an efficient representation and, importantly, how this accuracy, which is learned from imitation data, can be translated to the task priority. Second, we develop our stochastic combination approach using the task accuracy as relative priorities. Third, we show that both strict and "soft" prioritization approaches are special cases of our prioritization approach where some uncertainty parameters are set to zero.

To better illustrate our approach, we begin our description by prioritizing two controllers; an operational-space controller which has the highest priority and a low priority joint-space controller. Subsequently, we extend our approach to multiple operational-space controllers in Sec. III-C. We use operation-space prioritization as an illustration of our approach, but in Sec. IV we show that our stochastic prioritizing scheme can be generalized to a wider class of controllers.

A. Encoding Task Accuracy from Demonstrations

Representing the desired task accuracy throughout the duration of the task is critical for our approach. A measure for the task accuracy is the task variance that is be obtained over multiple executions of the task. Stochastic movement primitive representations can not only represent the task variance but also enable training from demonstration data. To this end, we use the Probabilistic Movement Primitives (ProMPs) approach [2], [3] as our representation.

ProMPs represent a single trajectory as a weighed linear combination of Gaussian basis functions $\boldsymbol{\Phi}_t$ and the respective weights $\boldsymbol{w}$, i.e.,

$$\boldsymbol{y}_t = \boldsymbol{\Phi}_t \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{y}_t = [\boldsymbol{x}, \dot{\boldsymbol{x}}]^T$ represents the state of the task, i.e., positions and velocities, at time t. The task state $\boldsymbol{y}_t$ is a vector that contains the variables that define the state of the tasks, e.g., the joint or end-effector positions and velocities. Each task demonstration is used to estimate the weights $\boldsymbol{w}$ for that execution using a maximum likelihood approach [2]. From the set of estimated weights, ProMPs estimate a distribution over the weights, i.e.,

$$p(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w} | \boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\right), \tag{2}$$

which is assumed to be approximated well by a Gaussian, or a mixture of Gaussian [25], [26]. Thus, ProMPs offer a compact representation of the trajectory distribution in task space, that is, the mean movement in task space, the correlation between the task's variables, and their variance. With ProMPs, we can evaluate the distribution of the state $p(\boldsymbol{y}_t)$ at every time-step

$$p(\boldsymbol{y}_t) = \int p(\boldsymbol{y}_t | \boldsymbol{w}) p(\boldsymbol{w}) \mathrm{d}\boldsymbol{w} = \mathcal{N}\left(\boldsymbol{y}_t | \boldsymbol{\mu}_{\boldsymbol{y}_t}, \boldsymbol{\Sigma}_{\boldsymbol{y}}\right) \tag{3}$$

in closed form. ProMPs also provide a stochastic linear controller, which is also derived in closed form. The controller can follow the encoded task distribution exactly, i.e., it matches mean and variance of the distribution. In [2], [3], ProMPs are used to control the joints of the robot and, therefore, the controller outputs are joint torques. In our approach, we generalize ProMPs to model and control task variables, e.g. the robot end-effector. To do so, we adjust the ProMP controller's output to the acceleration of task space variables. The stochastic controller is, therefore, given by

$$p(\ddot{\boldsymbol{x}} | \boldsymbol{y}_t) = \mathcal{N}\left(\ddot{\boldsymbol{x}} | \boldsymbol{K}_t \boldsymbol{y}_t + \boldsymbol{k}_t, \boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}}\right). \tag{4}$$

The mean of the controller is given by a linear feedback control law. The controller additionally contains the covariance of the task in the acceleration space. The later plays an important part in our approach as it specifies the required accuracy of the control, see Sec. III-B. In summary, ProMPs are capable of representing and learning the task covariance $\boldsymbol{\Sigma}_{\boldsymbol{y}}$, and transforming it to the acceleration covariance $\boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}}$.

B. Probabilistic Combination of Tasks

We begin our derivation given two tasks, a joint-space task and an operational-space task. For each task, a stochastic controller is obtained from the corresponding ProMP that has been trained from demonstrations. Each controller is normally distributed, i.e., the probability of each controller is given by

$$p_1(\ddot{\boldsymbol{q}}) \sim \mathcal{N}\left(\ddot{\boldsymbol{q}} | \boldsymbol{\mu}_{\ddot{\boldsymbol{q}}}, \boldsymbol{\Sigma}_{\ddot{\boldsymbol{q}}}\right), \quad p_2(\ddot{\boldsymbol{x}}) \sim \mathcal{N}\left(\ddot{\boldsymbol{x}} | \boldsymbol{\mu}_{\ddot{\boldsymbol{x}}}, \boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}}\right). \tag{5}$$

The vector $\ddot{\boldsymbol{q}}$ denotes the joint acceleration, for all of the joints of the robot and the vector $\ddot{\boldsymbol{x}}$ the operational-space acceleration. Equation (5) is true for every time-step, however, we dropped the time-index for simplicity.

The operational-space controller and the joint-space controller can not be used simultaneously without accounting for the kinematics of the system. The system kinematics introduce a constraint between the operational and the joint space acceleration. The constraint is commonly defined in the velocity space by $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}}$, where $\boldsymbol{J}$ denotes the Jacobian from a base-frame to the operational-space. Equivalently, by differentiation over time, we obtain the acceleration-space formulation $\ddot{\boldsymbol{x}} = \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$ of the constraint. The term $\dot{\boldsymbol{J}}$ denotes the time derivative of the Jacobian[1]. Given the constraint in the acceleration-space, the operational-space controller depends on the current joint-acceleration $\ddot{\boldsymbol{q}}$. The probability of the operational-space acceleration $\ddot{\boldsymbol{x}}$ given the joint acceleration $\ddot{\boldsymbol{q}}$ is given by the conditional

$$p_{2|\ddot{\boldsymbol{q}}}(\ddot{\boldsymbol{x}} | \ddot{\boldsymbol{q}}) \sim \mathcal{N}\left(\ddot{\boldsymbol{x}} \middle| \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}, \boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}}\right), \tag{6}$$

where the mean of the conditional distribution is given by the constraint and the variance is given by the desired task accuracy. We can now use the joint space ProMP as prior distribution and the desired task-space mapping $p_{2|\ddot{\boldsymbol{q}}}(\ddot{\boldsymbol{x}} = \boldsymbol{\mu}_{\ddot{\boldsymbol{x}}} | \ddot{\boldsymbol{q}})$ as likelihood to obtain the posterior distribution for the joint space controller using Bayes theorem, i.e.,

[1]The time derivative of the Jacobian $\boldsymbol{J}$ can be obtained by applying the chain-rule, i.e.,

$$\dot{\boldsymbol{J}} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial t}$$

$$p_{1|\ddot{x}}(\ddot{q}|\ddot{x}=\mu_{\ddot{x}}) = \frac{p_{2|\ddot{q}}(\ddot{x}=\mu_{\ddot{x}}|\ddot{q})p_1(\ddot{q})}{p_2(\ddot{x})} = \mathcal{N}\left(\ddot{q}|\mu,\Sigma\right), \tag{7}$$

where, since both the prior distribution $p(\ddot{q})$ and the conditional $p(\ddot{x}|\ddot{q})$ are Gaussian distributions, the posterior is also a Gaussian distribution. The control law for the joint accelerations $\ddot{q}$ is then obtained by computing the marginal distribution

$$p_{1|2}(\ddot{q}) = \int p_{1|\ddot{x}}(\ddot{q}|\ddot{x})p_2(\ddot{x})\mathrm{d}\ddot{x} = \mathcal{N}\left(\ddot{q}|\mu'_{\ddot{q}},\Sigma'_{\ddot{q}}\right), \tag{8}$$

that is a Gaussian as well. The mean $\mu'_{\ddot{q}}$ and the covariance $\Sigma'_{\ddot{q}}$ are computed analytically as

$$\mu'_{\ddot{q}} = J^\dagger\left(\mu_{\ddot{x}} - \dot{J}\dot{q}\right) + \left(I - J^\dagger J\right)\mu_{\ddot{q}} \tag{9}$$

$$\Sigma'_{\ddot{q}} = \left(I - J^\dagger J\right)\Sigma_{\ddot{q}} + J^\dagger \Sigma_{\ddot{x}} J^{\dagger,T} \tag{10}$$

where the $J^\dagger$ denotes the generalized inverse of the Jacobian

$$J^\dagger = \Sigma_{\ddot{q}} J^T\left(\Sigma_{\ddot{x}} + J\Sigma_{\ddot{q}}J^T\right)^{-1}. \tag{11}$$

In our approach, the joint space acceleration $\ddot{q}$ and the task-space acceleration $\ddot{x}$, are obtained from the stochastic feedback controller of the ProMPs. However, our approach can be modify to incorporate other stochastic feedback controllers as we evaluate in Sec. V-B. The variance of the operational-space controller $\Sigma_{\ddot{x}}$ is used as regularization matrix and the variance of the joint-space controller $\Sigma_{\ddot{q}}$ as weighting. The generalized inverse $J^\dagger$ is not a pseudo-inverse as $JJ^\dagger \neq I$, due to the regularization by the operational-space covariance $\Sigma_{\ddot{x}}$. Therefore, the matrix $\left(I - J^\dagger J\right)$ is not a proper null-space projection of the Jacobian J.

C. Extension to multiple tasks

Multiple operational-space controllers can be naturally integrated in our approach where each task $i \in 1 \cdots N$ can operate in a difference space. In principle, it is sufficient to compute the posterior distribution over the joint acceleration $\ddot{q}$, given the accelerations of all task controllers $\{\ddot{x}_i\}_{1\cdots N}$, i.e., $p(\ddot{q}|\{\ddot{x}_i\}_{1\cdots N})$, which can be computed recursively, or in a single step [1], where the single step solution require sparse-matrix inversion techniques for efficiency.

We proceed with the recursive computation. For the recursive computation, we begin with our prior distribution over the joint accelerations $p_1(\ddot{q})$. We condition it with the operational-space acceleration distribution $p_N(\ddot{x}_N)$ of the highest priority task. The resulting posterior distribution $p_{1|N}(\ddot{q}|\ddot{x}_N)$ is then used as a new prior distribution and is conditioned with $p_{N-1}(\ddot{x}_{N-1})$. We continue conditioning until we reach the task $i = 1$. During the computation of the new prior distribution at every step, we can perform a numerical stability analysis of the $(\Sigma_{\ddot{x}}+J\Sigma_{\ddot{q}}J^T)$ matrix inversion, e.g. by computing the condition number of the matrix. If the inversion becomes numerically unstable, then the task i_o added at this step is incompatible to the higher priority tasks $N \cdots i_o - 1$. Our recursive approach has similarities with the strict hierarchical prioritization approaches [5], [4], where they recursively project every lower-priority task in the null-space of the higher priority task. The major difference in our approach is the use of the regularized generalized inverse, as presented in Sec. III-B, We use the tasks accuracies obtained from imitation data, instead of treating each task with an infinite accuracy.

TABLE I
COMPARISON OF DIFFERENT PSEUDO INVERSES USED FOR OPERATIONS

$J^\dagger = J^T\left(JJ^T\right)^{-1}$	**Generalized inverse**
$J^\dagger = M^{-1}J^T\left(JM^{-1}J^T\right)^{-1}$	**Weighted generalized inverse**, weighted with the inverse of the mass
$J^\dagger = \Sigma_{\ddot{q}}J^T\left(\Sigma_{\ddot{x}} + J\Sigma_{\ddot{q}}J^T\right)^{-1}$	**Bayesian inverse**, weighting and regularization are computed in closed form.

IV. AN OPTIMIZATION POINT OF VIEW

We presented our derivation by prioritizing an operational-space controller and a joint-space controller, however, this was only a special case. Our approach can be generalized to a wider class of problems, where the constraints imposed are linear to the joint acceleration $\ddot{q}$, i.e. can be formulated as

$$A\ddot{q} = b, \tag{12}$$

where the matrix A and vector b possibly depend on the current state of the robot q and $\dot{q}$ at time t. The constraint imposed by the robot's mechanics can be re-formulated in the generalized form of Equation (12), by setting $A = J$ and $b = \ddot{x} - \dot{J}\dot{q}$.

We can now formulate a optimization problem that incorporates a soft version of this constraint while staying close to the prior mean. The covariance matrices serve as L2 norm metric for the objectives, i.e.,

$$\arg\min_{\ddot{q}} J = \arg\max_{\ddot{q}}\,(A\ddot{q} - b)^T\Sigma_{\ddot{x}}^{-1}(A\ddot{q} - b) + (\ddot{q} - \mu_{\ddot{q}})^T\Sigma_{\ddot{q}}^{-1}(\ddot{q} - \mu_{\ddot{q}})^T. \tag{13}$$

This formulation resembles the optimization framework presented in [4] with the difference that $A\ddot{q} - b$ is imposed as soft-constraint and not as hard constraint. If we let $\Sigma_{\ddot{x}}$ go to zero, we obtain a hard constraint and all the control laws in [4] can be recovered. However, the pseudo-inverse is lacking a proper regularization which leads to instabilities in singularities. Furthermore, the optimization view does not provide a direct way to update the joint covariance if several tasks need to be prioritized. In contrast, the joint covariance $\Sigma_{\ddot{q}}$ is updated in the Bayesian approach. It specifies the direction in joint space that violate all conditioned task space controllers as little as possible.

A. Comparison to Strict Prioritization Approaches.

Our control law can also be formulated for torques $\boldsymbol{u}$ instead of desired accelerations $\ddot{\boldsymbol{q}}$. These derivations are given in the appendix. We observe that the mean $\boldsymbol{\mu_u}$ of the controls, given in Equation (14), has a similar structure as well-known operational-space control laws [4], [5], [6], [7], [8], [9], [10], [11], [12]. It consists of a model-based component to compensate for the dynamics of the system, the desired acceleration in the operational-space —which, for example, can be the output of a feedback controller— and a projection component $\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)$.

The difference to the aforementioned approaches lays in the computation of the generalized inverse matrix of the Jacobian $\boldsymbol{J}^{\dagger}$. By applying a Bayesian approach, we obtain a generalized inverse matrix of the Jacobian which is both regularized and weighted, while strict prioritization methods use an un-regularized inverse.

The aforementioned approaches can be derived by assuming that the operational-space variance $\boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}}$ is zero, i.e. $\boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}} = \lim_{\alpha \to 0} \alpha \boldsymbol{I}$ and, therefore, degrade our approach to a deterministic case. If the operational-space variance is zero, the matrix $\boldsymbol{J}\boldsymbol{J}^{\dagger} = \boldsymbol{I}$ of the projection is a null-space projection, i.e. the lower priority tasks will not interfere with the higher priority tasks. Therefore, decreasing the variance of the operational-space controller $\boldsymbol{\Sigma}_{\ddot{\boldsymbol{x}}}$ can be interpreted as "hardening" the prioritization of the two controllers.

A consequence of not regularizing the generalized inverse is the numerical instability of the inversion at singular kinematic configurations. Some approaches [13], [14] suggest to add a small regularization of the form $\lambda \boldsymbol{I}$ to reduce the numerical instabilities. Performing such a regularization has the physical interpretation of adding a diagonal variance on accuracy of the high priority task. The projection $\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)$ will in this case not to be a null-space projection. However, to our knowledge, neither the motivation or the physical interpretation of this regularization has been previously discussed.

By additionally setting the joint-space covariance to $\boldsymbol{\Sigma}_{\ddot{\boldsymbol{q}}} = \boldsymbol{I}$, the pseudo inverse is un-regularized and unweighted and we obtain controls laws as in [4], [5], [6], [7], [8], [9], [10], [12]. Setting the joint-space covariance to $\boldsymbol{\Sigma}_{\ddot{\boldsymbol{q}}} = \boldsymbol{M}^{-1}$, we obtain controllers based on the Gauss principle of least constraint, and consistent to d'Alambert's principle of virtual work [11], [4]. The different approaches for computing the generalized inverse are shown in Table I.

V. Experimental Evaluation

We evaluate our approach on redundant simulated and physical robots performing tasks learned by imitation. As opposed to optimization approaches, our approach does not use a cost function, but learns the desired trajectory distribution from demonstrations. First, we demonstrate that our approach can be used for adapting known tasks, while the reproduction stays in the vicinity of the demonstrations. Second, we show that additional controllers can be smoothly integrated in our framework. Third, we build a library of tasks and show how we can use our approach to learn a combination of tasks with considerably improved data efficiency. We conclude the section by presenting our results on the humanoid platform "iCub" on initiating contacts and while reaching an object.

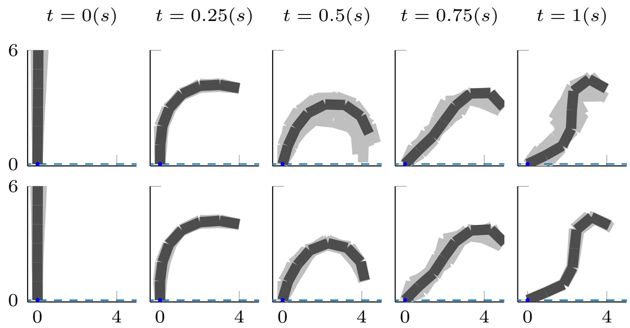


Fig. 2. A visualization of the 7-link planar robot trajectory for different time-steps. The dark configuration denotes the mean configuration. At every evaluation in time, we plot ten samples from the distribution to illustrate the variability of the movement. In the first row, we present the reproduction of the movement after training our approach. In the second row, we present the reproduction of the movement after conditioning at $t = 0.5(s)$ and $t = 1(s)$. We observe that the variance of the task-space movement reduces at the via-points.

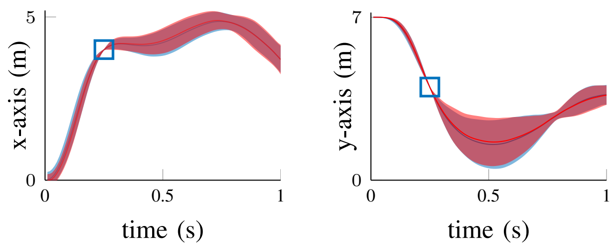


Fig. 3. We present the trajectory distribution of the end-effectors of the 7-link planar robot. The demonstrated trajectory distribution is shown in blue (blue line for the mean and shaded area for two times the standard deviation). The reproduced trajectory distribution that is obtained from our approach is shown in red. The reproduction distribution follows accurately the demonstrated one. The blue boxes illustrate a via-point (low variance of the movement at this time step) that was present during the demonstrations.

A. Data-Driven Task-Space Adaptation

In this experiment, we used a planar robot with seven Degrees of Freedom (DoF). We used optimal control to provide demonstrations to our approach. The demonstrations were provided in joint-space. We directly used the joint-space demonstrations for training a ProMP to be used as the lowest priority task. Additionally, we used the task-space trajectories to learn a task-space ProMP. The movement at different time steps is visualized in Figure 2. The demonstrations have different variability at different time-steps throughout the movement. Time step $t = 0.25s$ is a via-point, i.e. has very low variability, in both task-space dimensions. The demonstrated trajectory distribution in task-space is shown in Figure 3(blue), with the trajectory distribution obtained after reproduction(red). The reproduction distribution matches accurately the demonstrations and passes through the via-points. Additionally, we show that our approach can adapt a learned task. The adaptation is performed by using the "conditioning" operation of ProMPs on the task-space primitive. The prior ProMP is not changed by conditioning.

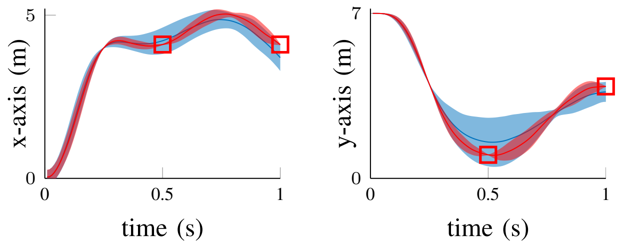


Fig. 4. The trajectory distribution of the end-effectors of the 7-link planar robot after conditioning. In blue, we present the trajectory distribution of the demonstrations. In red, the reproduction after conditioning only the task-space ProMP. The red boxes denote the additional via-points that were not present during the demonstrations. The reproduction can match both, the via-points of the demonstrations and the additional via-points, while it maintains the general shape of the movement. Our approach performs conditioning in task space while it stays close to the demonstrated data in joint space.

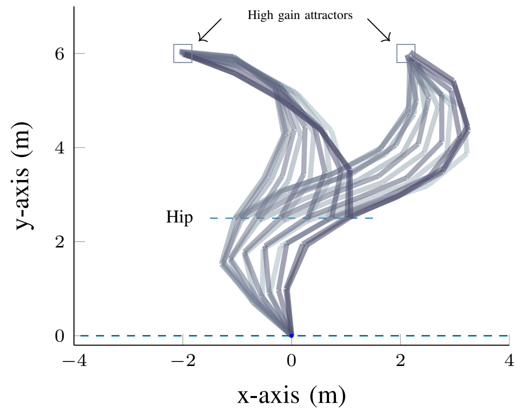


Fig. 5. The planar robot performing a bi-manually reaching task while moving its "hip". The robot accurately stays at the desired targets with its end-effectors during the movement. Additionally, the end of the "hip" link tracks a trajectory with a constant height.

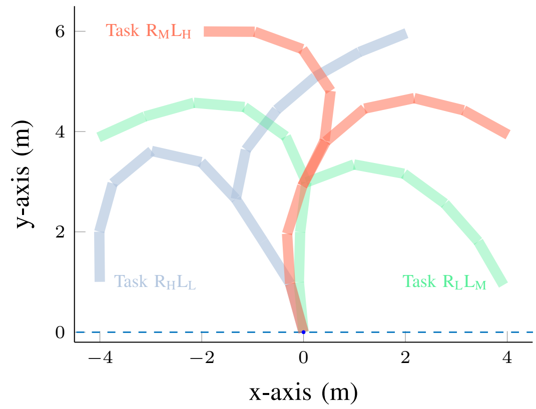


Fig. 6. A visualization of the two end-effector robot we used in our experiments. We present the final configuration, $t = t_{\text{end}}$ of the robot in task space for three different tasks, out of nine tasks we used in the experiment. Each color represents a different task combination. Our approach can learn the task of each end-effector, $\{R_*, L_*\}$, independently. The task of each end-effector, i.e., the reaching of a low point, a mid point, and a high point, is denoted by the L, M, H subscripts.

The adaptation does not require to run an inverse kinematics algorithm to find the respective joint configuration, but can be performed directly on the primitive and with a specified accuracy. We present our results in Figure 4, where we added two via-points to the movement. The reproduction can accurately pass through both via-points while it maintains the shape of the movement learned from the demonstrations.

B. Incorporation of External Control Laws

Furthermore, we present our results for a more complex planar robot with two end-effectors. The robot has three links for the torso and five additional links to represent arms. Each link is one meter long. The robot learned a movement where the "hip" moves in a constant height of $2.5m$. We show that expert-knowledge can be incorporated in our approach. The expert designed two feedback controllers with high gains and small variance $\mathbf{\Sigma}_{\boldsymbol{u}} = 10^3$ that attract the end-effectors at $\{2, 6\}(m)$ and $\{-2, 6\}(m)$, for the right and left end-effector respectively. The resulting movement is shown in Figure 5. The robot can perform all of the three tasks; it reproduces accurately the hip movement staying at the desired height and places its end-effectors at the desired locations set by the expert.

C. Combining Tasks of Different End-Effectors

In this evaluation we demonstrate the advantages of our approach a combination of tasks for different end-effectors. We use the planar robot with two end-effectors and thirteen DoF. First, we generated demonstrations where each end-effector has the task to reach one out of three end-points at $t_{\text{end}} = 1$. The end-point can either be "low", set at $\{4, 1\}$ for the right end-effector or at $\{-4, 1\}$ for the left, or "mid" at $\{\pm 4, 4\}$, or "high" at $\{\pm 2, 6\}$. The combination of all three tasks of the two end-effectors yields nine different task combinations. For each combination, we generate a set of noisy demonstration. We denote each task by $\{R_i, L_j\}$, where $\{R, L\}$ denotes the end-effector and $i, j \in \{L, M, H\}$ denotes the "low", "mid", or "high" end-point. An illustration of the configuration of the robot at these points is shown in Figure 6. As a baseline, we train nine individual primitives, one for each combination of tasks. However, our approach can use all available demonstrations per task of one end-effector, e.g. $\{L_M, R_*\}$, as it can learn the end-effector tasks independently resulting in a training set per task with three times the number of demonstrations as can be used for the baseline approach. Therefore, our approach considerably outperforms the baseline as the distributions can be estimated with higher accuracy. In Figure 7, we evaluate the average performance of both approaches which was specified as the negative square deviation from the true desired task-space position at the end of the movement. Our approach shows a superior accuracy due to the more efficient data usage. The trajectory distribution for both end-effectors is shown in Figure 8. In this Figure, we show that, using prioritization, we can also reproduce a task combination that has not been demonstrated to the robot.

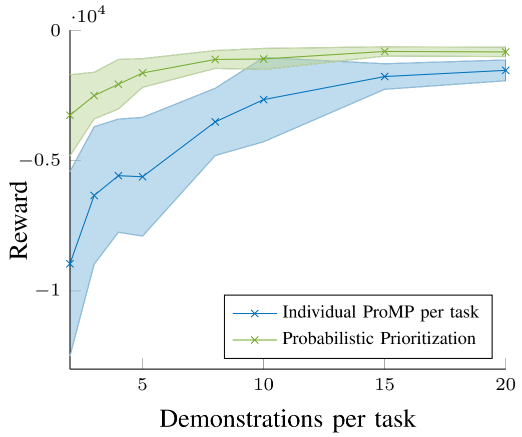


Fig. 7. Comparison between the Bayesian prioritization approach and learning each task independently which is used as a baseline. We vary the number of demonstrations used per task. Using prioritization, we can learn the tasks for each end-effector independently and therefore, can use more training data for the single tasks. For the baseline, we can only learn each combination of the end-effector tasks individually. The plot shows the average negative square deviation from the true desired end-effector position. Using prioritized primitives considerably improves the accuracy of learning due to the more efficient data usage.

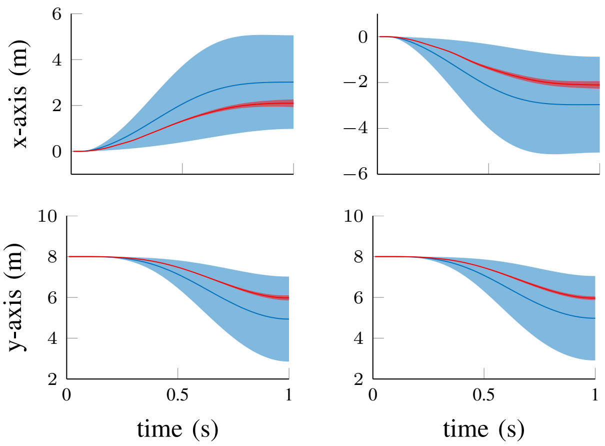


Fig. 8. The trajectory distribution of the end-effectors of the two end-effector robot. The first column presents the left end-effector and the second column of the right end-effector distributions. In blue, we show the prior distribution projected in Cartesian space after training with two task-combinations, $\{R_H, L_M\}$ and $\{R_M, L_H\}$. Our approach utilizes all available demonstrations. In red, we present the reproduction of of the $\{R_H, L_H\}$, a task combination that was not contained in the demonstrations, using our prioritization scheme.

D. Initiating Contacts during Reaching

In the final evaluation, we performed an experiment using the humanoid robot iCub to reach objects while improving its stability by partially supporting its weight on a table. The iCub was not mount at a pole, but was rather standing for the duration of our experiments Reaching objects that require the robot to bend the torso can move the center of gravity of the robot out of the support polygon defined by the feet, and, as a result, the robot will loose its balance. The task of the robot is to perform a reaching movement while it initiates a contact to stabilize the robot. The robot reaches for three

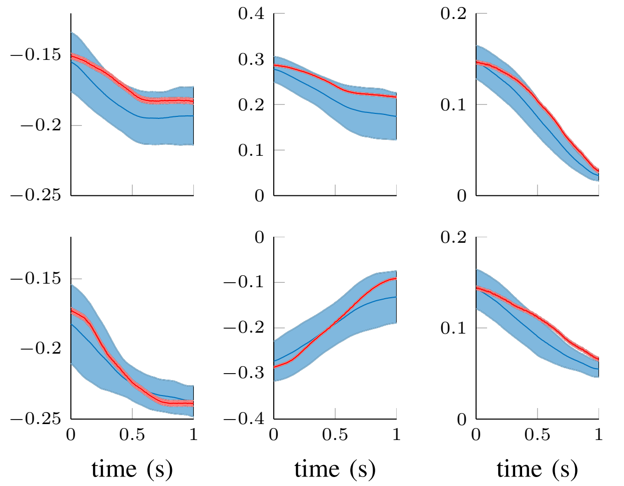


Fig. 9. The trajectory distribution of left (top) and right (bottom) of the iCub robot in Cartesian space, $\{x, y, z\}(m)$ shown in first, second, and third column respectively. We learned prior distribution of is shown in blue. During reproduction, we reached for a "blue—cake" combination (red). The variance of the reproduction distribution is due to the system noise, errors in the dynamics model, and friction.

different objects, as shown in Figure 1, with its right arm. Concurrently, with the left arm, it initiates a contact with the table that increases the stability of the robot. The location of the contact varies over three positions. We provided ten demonstrations reaching for different object locations and initiating different contacts with the forearm of the robot. The robot was capable of reproducing the movements using the prioritized movement primitives, as show in Figure 9. Additionally, the robot could perform unseen combinations of tasks and support locations.

VI. Conclusion

In this paper, we presented a novel approach for movement prioritization based on the combination of Bayesian task prioritization and the Probabilistic Movement Primitives. While prioritization is a well established concept in control, it has not been explored in the context of learning movement representations. We brought attention to the Bayesian task prioritization framework that allows for a principled treatment of the task priorities and avoids numerical instabilities. We combined it with the Probabilistic Movement primitives to enable learning the priorities from demonstrations.

In this paper, we have shown that combining prioritization with learning approaches yields in powerful representation that can be used to solve a combination of tasks with different end-effectors. Our approach is data-driven, i.e., it can solely be trained form demonstrations and minimizes expert knowledge. Especially, it avoids the problem of specifying a cost function for the task in hand, which is still an open problem. We demonstrated that our approach can be used to adapt task-space movements without solving an inverse kinematics problem and, importantly, staying close to the demonstrated data.

A key contribution of our approach is the ability to combine tasks of different end-effectors in a principle and data-efficient way. Our approach can generalize to task combinations that were not present in the demonstrations and requires significantly less training data to achieve the same level of performance.

In future work, we will expand the evaluations of our approach on more complex real-word scenarios. We consider multiple task execution with physical robot interaction under the present of contacts as interesting research direction.

APPENDIX
INCLUDING THE DYNAMICS OF THE SYSTEM

The stochastic controller on the joint acceleration given in Equation (7) can be used to control a physical system, i.e. by torque control, using the rigid-body dynamics model [27],

$$\boldsymbol{u} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}),$$

where $\boldsymbol{M}(\boldsymbol{q})$ denotes the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ denotes Coriolis and centripetal forces, and $\boldsymbol{G}(\boldsymbol{q})$ forces due to gravity. Using the rigid-body dynamics model, we reformulate our controller to operate in the joint torque space, i.e.

$$p_{1|2}(\boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{u}|\boldsymbol{\mu}'_{\boldsymbol{u}}, \boldsymbol{\Sigma}_{\boldsymbol{u}}\right).$$

The mean $\boldsymbol{\mu}_{\boldsymbol{u}}$ of this controller is given by

$$\begin{aligned}\boldsymbol{\mu}'_{\boldsymbol{u}} =& \boldsymbol{M}\left(\boldsymbol{J}^{\dagger}\left(\boldsymbol{\mu}_{\ddot{\boldsymbol{x}}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) + \left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)\boldsymbol{\mu}_{\ddot{\boldsymbol{q}}}\right) + \boldsymbol{C} + \boldsymbol{G} \\ =& \boldsymbol{M}\boldsymbol{J}^{\dagger}\left(\boldsymbol{\mu}_{\ddot{\boldsymbol{x}}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) \\ &+ \boldsymbol{M}\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)\left(\boldsymbol{M}^{-1}\left(\boldsymbol{\mu}_{\boldsymbol{u}} - \boldsymbol{C} - \boldsymbol{G}\right)\right) + \boldsymbol{C} + \boldsymbol{G},\end{aligned}$$

where we used $\boldsymbol{\mu}_{\ddot{\boldsymbol{q}}} = \boldsymbol{M}^{-1}(\boldsymbol{\mu}_{\boldsymbol{u}} - \boldsymbol{C} - \boldsymbol{G})$.

Furthermore, a decoupling of the kinematics and the dynamics can be obtained by setting $\hat{\boldsymbol{\mu}}_{\boldsymbol{u}} = \boldsymbol{\mu}_{\boldsymbol{u}} + \boldsymbol{C} + \boldsymbol{G}$ and using it in place of $\boldsymbol{\mu}_{\boldsymbol{u}}$. In this case, the mean becomes

$$\begin{aligned}\boldsymbol{\mu}'_{\boldsymbol{u}} =& \boldsymbol{M}\boldsymbol{J}^{\dagger}\left(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) \\ &+ \boldsymbol{M}\left(\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)\left(\boldsymbol{M}^{-1}\boldsymbol{\mu}_{\boldsymbol{u}}\right) + \boldsymbol{C} + \boldsymbol{G} \quad (14)\end{aligned}$$

which results in the resolved-acceleration controller [28], [29].

REFERENCES

[1] M. Toussaint and C. Goerick, "A bayesian view on motor control and planning," in *From Motor Learning to Interaction Learning in Robots*, 2010.

[2] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[3] A. Paraschos, G. Neumann, and J. Peters, "A probabilistic approach to robot trajectory generation," in *"Proceedings of the International Conference on Humanoid Robots (Humanoids)*, 2013.

[4] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal, "A unifying framework for robot control with redundant DOFs," *Autonomous Robots*, 2007.

[5] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Journal of Robotics and Automation*, 1987.

[6] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole-body dynamic behavior and control of human-like robots," *International Journal of Humanoid Robotics*, 2004.

[7] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-Priority based redundancy control of robot manipulators," *International Journal of Robotics Research (IJRR)*, 1987.

[8] L. Sentis and O. Khatib, "Synthesis of Whole-Body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, 2005.

[9] ——, "A whole-body control framework for humanoids operating in human environments," in *International Conference on Robotics and Automation (ICRA)*, 2006.

[10] J. Park, W.-K. Chung, and Y. Youm, "Characterization of instability of dynamic control for kinematically redundant manipulators," in *International Conference on Robotics and Automation (ICRA)*, 2002.

[11] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," in *International Conference on Robotics and Automation (ICRA)*, 2000.

[12] J. Luh, M. Walker, and R. Paul, "Resolved-acceleration control of mechanical manipulators," *Transactions on Automatic Control*, 1980.

[13] P. Baerlocher and R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures," in *International Conference on Intelligent Robots and Systems (IROS)*, 1998.

[14] ——, "An inverse kinematics architecture enforcing an arbitrary number of strict priority levels," *The Visual Computer*, 2004.

[15] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions," in *International Conference on Robotics and Automation (ICRA)*, 2011.

[16] J. Salini, S. Barthélemy, P. Bidaud, and V. Padois, "Whole-Body motion synthesis with LQP-Based controller – application to icub," in *Modeling, Simulation and Optimization of Bipedal Walking*, 2013.

[17] W. Decre, R. Smits, H. Bruyninckx, and J. D. Schutter, "Extending iTaSC to support inequality constraints and non-instantaneous task specification," in *International Conference on Robotics and Automation (ICRA)*, 2009.

[18] R. Lober, V. Padois, and O. Sigaud, "Multiple task optimization using dynamical movement primitives for whole-body reactive control," in *International Conference on Humanoid Robots (Humanoids)*, 2014.

[19] ——, "Variance modulated task prioritization in Whole-Body control," in *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[20] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi, "Learning soft task priorities for control of redundant robots," in *International Conference on Robotics and Automation (ICRA)*, 2016.

[21] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[22] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, Nov. 2012.

[23] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *Transactions on Robotics*, 2011.

[24] L. Rozo, S. Calinon, D. Caldwell, P. Jiménez, and C. Torras, "Learning collaborative impedance-based robot behaviors," in *AAAI Conference on Artificial Intelligence*, 2013.

[25] M. Ewerton, G. Neumann, R. Lioutikov, H. Ben Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *International Conference on Robotics and Automation (ICRA)*.

[26] E. Rueckert, J. Mundo, A. Paraschos, J. Peters, and G. Neumann, "Extracting Low-Dimensional control variables for movement primitives," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015.

[27] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[28] T. Yoshikawa, *Foundations of robotics: analysis and control*. Mit Press, 1990.

[29] P. Hsu, J. Hauser, and S. Sastry, "Dynamic control of redundant manipulators," in *International Conference on Robotics and Automation (ICRA)*, 1988.

Bibliography

[1] R. Lober, V. Padois, and O. Sigaud. Multiple task optimization using dynamical movement primitives for whole-body reactive control. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Madrid, Spain, 2014.

[2] R. Lober, V. Padois, and O. Sigaud. Variance modulated task prioritization in whole-body control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3944–3949, Hamburg, Germany, Sep 2015.

[3] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi. Learning soft task priorities for control of redundant robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2016.

[4] A. Paraschos, J. Peters, and G. Neumann. Probabilistic prioritization of movement primitives, 2017. under review.