Learning to Sequence and Blend Robot Skills via Differentiable Optimization

Noémie Jaquier, You Zhou, Julia Starke, and Tamim Asfour

***Abstract*— In contrast to humans and animals who naturally execute seamless motions, learning and smoothly executing sequences of actions remains a challenge in robotics. This paper introduces a novel skill-agnostic framework that learns to sequence and blend skills based on differentiable optimization. Our approach encodes sequences of previously-defined skills as quadratic programs (QP), whose parameters determine the relative importance of skills along the task. Seamless skill sequences are then learned from demonstrations by exploiting differentiable optimization layers and a tailored loss formulated from the QP optimality conditions. Via the use of differentiable optimization, our work offers novel perspectives on multitask control. We validate our approach in a pick-and-place scenario with planar robots, a pouring experiment with a real humanoid robot, and a bimanual sweeping task with a human model.**

I. Introduction

Humans and animals generally achieve seamless sequences of actions, featuring smooth and natural transitions. Indeed, there are biological evidences that motor actions are composed of fundamental building blocks, which are then smoothly sequenced and combined to realize complex motions [1], [2]. This particularly applies to manipulation tasks, which can be broken down into several smoothly-linked action phases for which the brain selects and executes appropriate controllers [3]. In contrast, learning and executing seamless sequences of actions is still a challenge in robotics. Indeed, skills are usually learned for a specific task and are thus difficult to re-use in a different sequence of actions. Moreover, robot motions are characterized by obvious jerky transitions, which are so typical that people imitate robots by introducing abrupt pauses between subsequent movements.

In this paper, we propose a novel skill-agnostic approach to sequence and blend skills. To do so, we encode sequences of skills as quadratic programs (QP) [4] and leverage differentiable optimization (Optnet) layers [5], [6] to determine the relative importance of each skill throughout the task (see § III for a background). Our approach is skill-agnostic by acting on a set of control values, thus considering skills as a-priori given black-box solutions. Given a set of previously-defined (i.e., learned or programmed) skills and few demonstrations of a task, our formulation not only learns a suitable sequence of possibly-concurrent skills, but also blends transitions "for free", i.e., requiring no additional operations (see § IV).

The contributions of this paper are: (*i*) We propose a novel QP-based approach to learn seamless sequences of skills from demonstrations; (*ii*) We formulate a tailored loss function from the optimality of the QP; (*iii*) We present two types of QP parameters to encode the importance of skills; (*iv*) We bring a novel perspective on multitask control via the use of differentiable optimization. We showcase our approach in various experiments with simulated and real robots (§ V).

This work was supported by the Helmholtz AI project LearnGraspPhases and the Carl Zeiss Foundation through the JuBot project. The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany. Correspondence to: {noemie.jaquier, asfour}@kit.edu

II. Related Work

Given a set of individual robotic skills, the challenge is to order and combine them to successfully execute complex manipulation tasks. Sequencing approaches presented in the literature are mainly based on learning from demonstrations (LfD) [7], [8], [9], [10] or on reinforcement learning (RL) [10], [11]. Manschitz *et al.* [7] learn both a sequence graph of skills from demonstrations, and a classifier to select the transitions. The authors extend their approach to handle concurrent skill activations [8]. As opposed to our work, the transitions between skills are explicitly labeled for the demonstrations. Rozo *et al.* [9] introduce an object-centered skill sequencing formulation, which builds a complete model of the task by cascading several skill models, and adapting their task parameters. In contrast to our approach, the desired skill sequence is assumed to be given. In [10], demonstrated trajectories are segmented into sequences of skills, where skill policies are represented by linear value function approximations. Sequences from several demonstrations are then combined into skill trees. Stulp *et al.* [11] extend the PI^2 algorithm to optimize sequences of dynamical movement primitives (DMP) by simultaneously learning their shape and goal parameters. Overall, the aforementioned approaches are specifically tailored to a single skill type, e.g., dynamical systems [7], [8], task-parametrized Gaussian mixture model (TP-GMM) [9], or DMP [11]. Moreover, transitions are usually handled by matching the end- and start-points of subsequent skills, and are thus characterized by obvious pauses. In contrast, our approach is *skill-agnostic* and learns sequences featuring *seamless and natural* transitions.

Other works focus on designing smooth transitions between skills. For instance, several approaches were presented in [12] to blend DMPs, and probabilistic movement primitives (ProMP) can naturally be blended [13]. However, these methods require a known sequence of specific skills and a manual tuning of transition parameters. In [14], motions are generated from a hierarchy of motion primitives, which are activated based on a neural-like dynamics. Therefore, sequencing and blending is achieved by choosing suitable weights and connections. This approach was then combined with optimal control for continuous motion adaptation [15].

Although it generates seamless motions, its applicability is limited due to the necessity of defining the model by hand.

Sequencing and blending of tasks has also been explored in the context of robot multitask control. Salini *et al.* [16] combine different controllers in a QP formulation by defining a soft hierarchy of tasks. This corresponds to defining a sequence of skills with concurrent activations. Smooth transitions are achieved by smoothly-varying the relative importance of skills (priorities) with manually-tuned weights. In [17], the skills priorities are instead optimized using covariance matrix adaptation evolution strategy (CMA-ES) in order to superpose several controllers for motion generation. Modugno *et al.* [18] extended this idea to learn time-varying skill priorities given as a weighted sum of basis functions equally spaced in time. The corresponding weights can then be optimized using black-box optimization techniques such as CMA-ES [18] or Bayesian optimization (BO) [19], [20]. Our work distinguishes in that we directly learn the relative importance of skills along the task by *differentiating* through the optimization problem. In contrast to [18], [19], [20], we leverage LfD to learn sequences of previously-defined skills with seamless transitions. Therefore, our approach requires only few initial demonstrations and no additional trials during the learning phase, thus improving on data-efficiency and training cost compared to black-box optimization techniques.

III. Background

A. Multitask control with quadratic programming

Quadratic programs (QP) [4, Chap. 16] are extensively used to formulate multitask control of humanoid robots as a constrained optimization problem. Indeed, QP can be solved very efficiently, while explicitly incorporating a wide variety of objectives and accounting for diverse constraints (see e.g., [21], [22]). A QP solves a problem of the form

$$\min_{\boldsymbol{z}} \tfrac{1}{2}\boldsymbol{z}^\mathsf{T}\boldsymbol{Q}\boldsymbol{z} + \boldsymbol{c}^\mathsf{T}\boldsymbol{z} \;\text{ s. t. }\; \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b} \text{ and } \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{h}, \qquad (1)$$

where $\boldsymbol{z} \in \mathbb{R}^n$ is the optimization variable, $\boldsymbol{Q} \in \mathcal{S}_+^n$, $\boldsymbol{c} \in \mathbb{R}^n$ are the parameters of the quadratic cost function with $\mathcal{S}_+^n$ denoting the manifold of positive-semidefinite (PSD) matrices, and $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{G} \in \mathbb{R}^{p\times n}$, $\boldsymbol{h} \in \mathbb{R}^p$ are the constraints parameters. For robot multitask control, QP are typically used to minimize the weighted sum of a set of K tasks, i.e., $\min_{\boldsymbol{\xi}_1 \dots \boldsymbol{\xi}_K} \sum_{k=1}^K w_k \|\hat{\boldsymbol{\xi}}_k - \boldsymbol{\xi}_k\|^2$, where $\hat{\boldsymbol{\xi}}_k$ and $\boldsymbol{\xi}_k$ are the desired and current value of the task k, respectively, and w_k is a weight setting the relative importance of the task k with respect to the other tasks. Moreover, the constraints typically include the equations of motion (kinematics, or dynamics), the technological limits of the system (e.g., joint limits), and interaction constraints (e.g., grasp or frictional contacts). In this paper, we use a QP to encode a sequence of skills, along which the weights scaling the importance of each skill vary, leading to smooth trajectories and transitions.

B. Karush-Kuhn-Tucker conditions

The Karush-Kuhn-Tucker (KKT) conditions [23] are first order necessary conditions for $\boldsymbol{z}^*$ to be a local solution of a constrained optimization problem. In particular, the KKT conditions corresponding to the QP (1) are (*i*) $\nabla_{\boldsymbol{z}}\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\nu}) = 0$ with $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\nu})$ the Lagrangian function of the problem (1), and $\boldsymbol{\mu}, \boldsymbol{\nu}$ the Lagrangian multipliers corresponding to its equality and inequality constraints, respectively, (*ii*) $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}$, (*iii*) $\boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{h}$, (*iv*) $\mu_i \geq 0\ \forall i \in \{1 \dots m\}$, and (*v*) $\nu_j \geq 0\ \forall j \in \{1 \dots p\}$.

In addition to being used throughout the solving process of constrained optimization problems, the KKT conditions were exploited in inverse optimal control (IOC). In IOC, trajectories are viewed as the solution of an optimization problem, which aims at minimizing an unknown (parametric) cost. In this context, Englert *et al.* [24] used the fact that demonstrations of such trajectories — under the assumption that they are optimal — fulfill the KKT conditions, to determine the optimal parameters of the underlying cost.[1] We follow a similar reasoning and leverage the QP KKT conditions to define the loss of our sequencing approach.

C. Differentiable optimization layers

Recent works [5], [6] proposed to integrate optimization layers into neural architectures by differentiating through the corresponding optimization problems. In particular, Amos and Kolter [5] introduced *Optnet*, a neural architecture embedding QP as individual layers. Namely, Optnet defines the output $\boldsymbol{z}_{i+1}$ of the current layer as the solution of a QP whose parameters depend on the previous layer $\boldsymbol{z}_i$, i.e.,

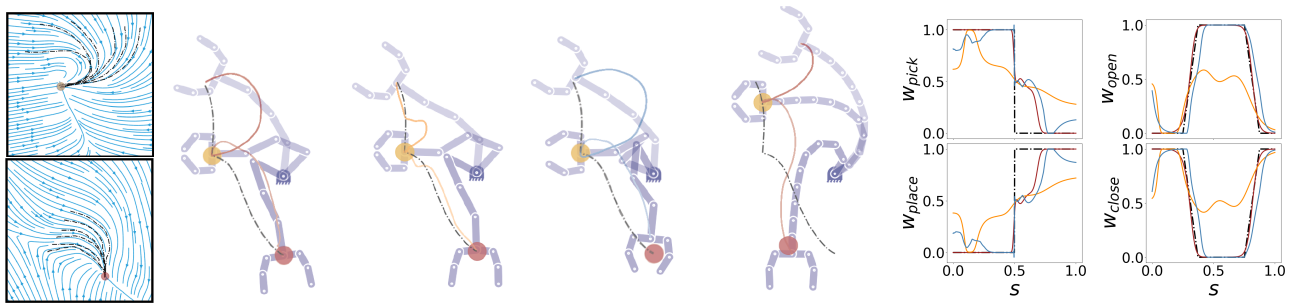
$$\begin{aligned}\boldsymbol{z}_{i+1} = \min_{\boldsymbol{z}} \tfrac{1}{2}\boldsymbol{z}^\mathsf{T}\boldsymbol{Q}(\boldsymbol{z}_i)\boldsymbol{z} + \boldsymbol{c}(\boldsymbol{z}_i)^\mathsf{T}\boldsymbol{z} \\ \text{s. t. } \boldsymbol{A}(\boldsymbol{z}_i)\boldsymbol{z} = \boldsymbol{b}(\boldsymbol{z}_i) \text{ and } \boldsymbol{G}(\boldsymbol{z}_i)\boldsymbol{z} \leq \boldsymbol{h}(\boldsymbol{z}_i). \qquad (2)\end{aligned}$$

In order to train Optnet using backpropagation, the layer (2) must be differentiable, i.e., the derivatives of the solution $\boldsymbol{z}_{i+1}$ of the QP with respect to its input parameters $\{\boldsymbol{Q}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{G}, \boldsymbol{h}\}(\boldsymbol{z}_i)$ must be computed. This is achieved by differentiating the KKT conditions of the problem at a given solution (see [5]). In this paper, we leverage Optnet to learn the importance of individual skills throughout the task.

IV. Learning to Sequence and Blend Skills

In this section, we present our approach to sequence and blend manipulation skills. In the following, we assume a set of previously-defined individual robot skills $\{\mathsf{C}_k\}_{k=1}^K$ (e.g., a skill library). The skills are considered as given black-box solutions, implying that their representations are unknown and may differ across the skills. At each instant, each skill outputs a desired control value $\hat{\boldsymbol{\xi}}_k(\boldsymbol{\psi})$, depending on a current state $\boldsymbol{\psi}$, to be given to the robot in order to execute the skill. For example, dynamical-systems-based skills [26] provide a desired end-effector velocity depending on the current end-effector position, and time-dependent skills such as [27] may output, e.g., a time-varying desired joint or end-effector position. The control values are specific to and may differ across skills. We then consider a manipulation task consisting of an *unknown* sequence of (some of) the

[1] Similar ideas have also been explored in the context of inverse reinforcement learning (IRL), where the parameters of a reward function were selected by minimizing the norm of the expert's policy gradient [25].



(a) $\mathsf{C}_{\mathsf{pick}}, \mathsf{C}_{\mathsf{place}}$ (b) Diag. $\boldsymbol{W}(s)$ (c) Full $\boldsymbol{W}(s)$ (d) Baseline $\boldsymbol{W}$ (e) Generalization (f) Evolution of diag$(\boldsymbol{W}(s))$

Fig. 1: Pick(●)-and-place(●) task with planar robots. *(a)* Pick (*top*) and place (*bottom*) DS skills (→). *(b)-(f)* Demonstration (- -), reproduction with the 4-DoF robot using diagonal (—), full (—) and baseline (—) weights, and generalization with the 10-DoF robot.

aforementioned skills, possibly concurrently activated. We observe one or several *optimal* demonstrations $\{\tilde{\boldsymbol{\tau}}^{(d)}\}_{d=1}^{D}$ of the task consisting of the observed control values, i.e., $\tilde{\boldsymbol{\tau}}^{(d)} = [\{\tilde{\boldsymbol{\xi}}_k^{(d)}(\boldsymbol{\psi}_s)\}_{k=1}^{K}]_{s=0}^{1}$, where the phase variable $s \in [0,1]$ encodes the task progress[2]. In other words, $s=0$ and $s=1$ represent the beginning and the end of the task.

A. Illustrative example: pick-and-place with planar robots

For the sake of clarity of this section, the different concepts underlying our approach are introduced generally before being illustrated for a pick-and-place task executed by planar robots with grippers. In this example, we observe a *single* manually-designed demonstration $\tilde{\boldsymbol{\tau}}^{(1)}$ provided by a 4-DoF teacher robot that picks an object, transports it, and places it at a given location (see Fig. 1). The demonstration steps were achieved with proportional controllers activated using the weights of Fig. 1f. We then consider a set of four skills $\{\mathsf{C}_{\mathsf{pick}}, \mathsf{C}_{\mathsf{place}}, \mathsf{C}_{\mathsf{open}}, \mathsf{C}_{\mathsf{close}}\}$, where the pick/place and open/close skills control the arm and gripper motion, respectively. Although we next disclose the skills types, remember that they are considered as given black-box solutions in our approach. Indeed, each skill only provides a desired control value $\hat{\boldsymbol{\xi}}_k(\boldsymbol{\psi}_s)$ depending on the state $\boldsymbol{\psi}_s$ at each task instant.

The arm skills are encoded as dynamical systems (DS) [26] trained with the control Lyapunov function scheme of [28]. The obtained DS, illustrated by Fig. 1a, can then be adapted to new situations via translations and rotations. The desired control values of the DS-based skills correspond to the end-effector velocity $\dot{\boldsymbol{p}}$ and depend on the current end-effector position $\boldsymbol{p}_s$, such that $\hat{\boldsymbol{\xi}}_{\mathsf{pick}}(\boldsymbol{\psi}_s) \equiv \hat{\dot{\boldsymbol{p}}}_{\mathsf{pick}}(\boldsymbol{p}_s)$ and $\hat{\boldsymbol{\xi}}_{\mathsf{place}}(\boldsymbol{\psi}_s) \equiv \hat{\dot{\boldsymbol{p}}}_{\mathsf{place}}(\boldsymbol{p}_s)$. The desired control values of the gripper skills correspond to the velocity of the gripper joints $\dot{\gamma}$. The velocities $\hat{\dot{\gamma}}_{\mathsf{open}}$ and $\hat{\dot{\gamma}}_{\mathsf{close}}$ are zero when the gripper is completely opened or closed, and constant otherwise, i.e., $\hat{\boldsymbol{\xi}}_{\mathsf{open}}(\boldsymbol{\psi}_s) \equiv \hat{\dot{\gamma}}_{\mathsf{open}}(\gamma_s)$ and $\hat{\boldsymbol{\xi}}_{\mathsf{close}}(\boldsymbol{\psi}_s) \equiv \hat{\dot{\gamma}}_{\mathsf{close}}(\gamma_s)$. In this example, the phase variable s is defined as $s = t/T$ with t the elapsed time, and T the total duration of the task.

B. Sequencing and blending of skills with QPs

Similarly to multitask control, we propose to encode sequences of skills as QPs. Namely, given the desired control values $\{\hat{\boldsymbol{\xi}}_k\}_{k=1}^{K}$ output by the K individual skills and the current control values $\{\boldsymbol{\xi}_k\}_{k=1}^{K}$, a sequence of skills can be generated by solving the following optimization problem

$$\min_{\{\boldsymbol{\xi}_k\}_{k=1}^{K}} \frac{1}{2} \begin{pmatrix} \hat{\boldsymbol{\xi}}_1 - \boldsymbol{\xi}_1 \\ \vdots \\ \hat{\boldsymbol{\xi}}_K - \boldsymbol{\xi}_K \end{pmatrix}^{\mathsf{T}} \boldsymbol{W}(s) \begin{pmatrix} \hat{\boldsymbol{\xi}}_1 - \boldsymbol{\xi}_1 \\ \vdots \\ \hat{\boldsymbol{\xi}}_K - \boldsymbol{\xi}_K \end{pmatrix}, \tag{3}$$

at each $s \in [0,1]$, where $\boldsymbol{W}(s)$ is a varying weight matrix setting the relative importance of the skills throughout the sequence in function of the phase variable s encoding the task progress. The problem (3) is usually augmented with linear constraints related to the robotic system (see § III-A). In our case, we also include equality constraints for control values of the same type, i.e., $\boldsymbol{\xi}_i = \boldsymbol{\xi}_j$ if the skills i and j have the same type of outputs (e.g., both return end-effector pose values). For instance, following (3), the optimization problem of our illustrative example is formulated as

$$\min_{\{\dot{\boldsymbol{p}}_{\mathsf{pick}}, \dot{\boldsymbol{p}}_{\mathsf{place}}, \dot{\gamma}_{\mathsf{open}}, \dot{\gamma}_{\mathsf{close}}\}} \frac{1}{2} \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}_{\mathsf{pick}} - \dot{\boldsymbol{p}}_{\mathsf{pick}} \\ \hat{\dot{\boldsymbol{p}}}_{\mathsf{place}} - \dot{\boldsymbol{p}}_{\mathsf{place}} \\ \hat{\dot{\gamma}}_{\mathsf{open}} - \dot{\gamma}_{\mathsf{open}} \\ \hat{\dot{\gamma}}_{\mathsf{close}} - \dot{\gamma}_{\mathsf{close}} \end{pmatrix}^{\mathsf{T}} \boldsymbol{W}(s) \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}_{\mathsf{pick}} - \dot{\boldsymbol{p}}_{\mathsf{pick}} \\ \hat{\dot{\boldsymbol{p}}}_{\mathsf{place}} - \dot{\boldsymbol{p}}_{\mathsf{place}} \\ \hat{\dot{\gamma}}_{\mathsf{open}} - \dot{\gamma}_{\mathsf{open}} \\ \hat{\dot{\gamma}}_{\mathsf{close}} - \dot{\gamma}_{\mathsf{close}} \end{pmatrix},$$
$$\text{s.t. } \dot{\boldsymbol{p}}_{\mathsf{pick}} = \dot{\boldsymbol{p}}_{\mathsf{place}} \text{ and } \dot{\gamma}_{\mathsf{open}} = \dot{\gamma}_{\mathsf{close}}.$$

The constraints come from the shared control values across skills, i.e., the end-effector velocity $\dot{\boldsymbol{p}}$, and the gripper joints velocity $\dot{\gamma}$ for the pick/place and open/close skills, respectively. These constraints can directly be integrated into the optimization problem, which is equivalently written as

$$\min_{\{\dot{\boldsymbol{p}}, \dot{\gamma}\}} \frac{1}{2} \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}_{\mathsf{pick}} - \dot{\boldsymbol{p}} \\ \hat{\dot{\boldsymbol{p}}}_{\mathsf{place}} - \dot{\boldsymbol{p}} \\ \hat{\dot{\gamma}}_{\mathsf{open}} - \dot{\gamma} \\ \hat{\dot{\gamma}}_{\mathsf{close}} - \dot{\gamma} \end{pmatrix}^{\mathsf{T}} \boldsymbol{W}(s) \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}_{\mathsf{pick}} - \dot{\boldsymbol{p}} \\ \hat{\dot{\boldsymbol{p}}}_{\mathsf{place}} - \dot{\boldsymbol{p}} \\ \hat{\dot{\gamma}}_{\mathsf{open}} - \dot{\gamma} \\ \hat{\dot{\gamma}}_{\mathsf{close}} - \dot{\gamma} \end{pmatrix}. \tag{4}$$

Note that (3) can be equivalently formulated as (1) with the optimization variable $\boldsymbol{z} = (\boldsymbol{\xi}_1^{\mathsf{T}} \dots \boldsymbol{\xi}_K^{\mathsf{T}})^{\mathsf{T}}$, and cost parameters $\boldsymbol{Q} = \boldsymbol{W}$, $\boldsymbol{c} = -\boldsymbol{W}\hat{\boldsymbol{z}}$ with $\hat{\boldsymbol{z}} = (\hat{\boldsymbol{\xi}}_1^{\mathsf{T}} \dots \hat{\boldsymbol{\xi}}_K^{\mathsf{T}})^{\mathsf{T}}$. Importantly, the skill ordering in (3) is arbitrary. Indeed, the sequence is defined by the weight matrix, that is learned from demonstrations, as explained next. Skills can be added by extending $\hat{\boldsymbol{z}}$ with their control values and expanding $\boldsymbol{W}$ accordingly.

Given one or several D demonstrations $\{\tilde{\boldsymbol{\tau}}^{(d)}\}_{d=1}^{D}$ of a manipulation task, we aim at learning the skill weight function $s \mapsto \boldsymbol{W}(s) : \mathbb{R} \to \mathcal{S}_+^n$, so that the reproduction $\boldsymbol{\tau} = [\{\boldsymbol{\xi}_{k,s}^*\}_{k=1}^{K}]_{s=0}^{1}$, i.e., the sequence of skills

[2]In the remainder we drop dependencies on $\boldsymbol{\psi}_s$ to simplify the notation.

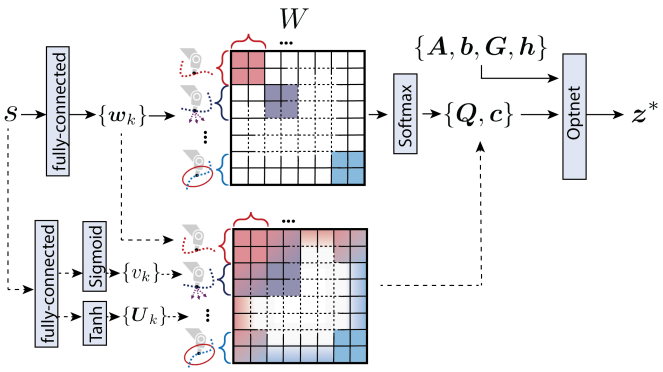


Fig. 2: Illustration of the proposed learning approach. The relative importance of the skills is encoded by $\boldsymbol{W}$ as a function of s. An Optnet layer, solving a QP whose parameters depend on $\boldsymbol{W}$, is then used to determine the control command $\boldsymbol{z}^*$. $\boldsymbol{W}$ is either a block-diagonal (*top*), or a full (*bottom*) matrix. The dashed arrows are only activated in the latter to learn the off-diagonal elements.

obtained by solving (3) for $s \in [0, 1]$, replicates the demonstrated task. This corresponds to minimizing a loss function $\ell(\boldsymbol{\tau}, \{\tilde{\boldsymbol{\tau}}^{(d)}\}_{d=1}^{D})$ measuring the quality of the reproduction. To do so, we need to solve a nested optimization: For each time instance of the task, we solve (3), and the whole set of solutions $[\{\boldsymbol{\xi}_{k,s}^*\}_{k=1}^{K}]_{s=0}^{1}$ is then used to minimize the loss $\ell(\boldsymbol{\tau}, \{\tilde{\boldsymbol{\tau}}^{(d)}\}_{d=1}^{D})$. To solve this problem, we leverage Optnet [5] to integrate the QP (3) into a neural network. Optnet allows us (*i*) to represent the QP parameters as functions, and (*ii*) to differentiate ℓ with respect to the QP parameters to solve the outer optimization of our nested problem using gradient-based approaches. In other words, Optnet backpropagates the loss ℓ to optimize both the phase-dependent skills weights $\boldsymbol{W}(s)$ and the control outputs $\boldsymbol{z}$. Thus, we can learn the relative importance of the skills throughout the task execution via the matrix $\boldsymbol{W}(s)$. Our proposed neural network takes the phase variable s as input, and consists of (*i*) a fully-connected layer coupled with a softmax activation function, whose outputs are the QP parameters $\{\boldsymbol{Q}, \boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{G}, \boldsymbol{h}\}(s)$ (see § IV-D for details), and (*ii*) of an Optnet layer (2), where $\boldsymbol{z}_i = s$, and $\boldsymbol{z}_{i+1} = \boldsymbol{z}*$ is the control command transmitted to the robot to execute the task. Our approach is illustrated by Fig. 2.

It is important to emphasize that the proposed formulation not only learns sequences of skills, but also blends the transition between individual skills "for free". Indeed, the coupling of the fully-connected layer with a softmax activation induces smooth non-binary weight functions $\boldsymbol{W}(s)$, therefore leading to smooth variations of the relative importance of the skills, i.e., to smooth transitions. This allows our neural architecture to learn and reproduce seamless transitions, as usually observed in human demonstrations. This also implies that skills are not necessarily executed in a strict sequence, but may be activated concurrently if required by the task.

The individual skills outputs $\dot{\boldsymbol{\xi}}_k$ may be defined either in task space (e.g., end-effector pose, or velocity), or in joint space (e.g., joint position, or velocity). In the former case, it may be desirable to directly solve the optimization (3) with respect to joint variables when executing the reproduction on the robot. To do so, the current control values $\{\boldsymbol{\xi}_k\}_{k=1}^{K}$ can be expressed in function of the joint values by exploiting the kinematic or dynamic relationship between the task- and joint-space variables. In our illustrative example, this corresponds to solving, during the reproduction,

$$\min_{\{\dot{\boldsymbol{\alpha}}, \dot{\gamma}\}} \frac{1}{2} \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}_{\text{pick}} - \boldsymbol{J}\dot{\boldsymbol{\alpha}} \\ \hat{\dot{\boldsymbol{p}}}_{\text{place}} - \boldsymbol{J}\dot{\boldsymbol{\alpha}} \\ \hat{\dot{\gamma}}_{\text{open}} - \dot{\gamma} \\ \hat{\dot{\gamma}}_{\text{close}} - \dot{\gamma} \end{pmatrix}^{\mathsf{T}} \boldsymbol{W}(s) \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}_{\text{pick}} - \boldsymbol{J}\dot{\boldsymbol{\alpha}} \\ \hat{\dot{\boldsymbol{p}}}_{\text{place}} - \boldsymbol{J}\dot{\boldsymbol{\alpha}} \\ \hat{\dot{\gamma}}_{\text{open}} - \dot{\gamma} \\ \hat{\dot{\gamma}}_{\text{close}} - \dot{\gamma} \end{pmatrix}, \tag{5}$$

where the arm skills outputs are expressed as $\dot{\boldsymbol{p}} = \boldsymbol{J}\dot{\boldsymbol{\alpha}}$ with $\dot{\boldsymbol{\alpha}}$ and $\dot{\gamma}$ the arm and gripper joint velocities, respectively, and $\boldsymbol{J}$ the manipulator Jacobian. Finally, note that nonlinear relationships must be linearized for the QP formulation.

C. Definition of the loss function

In this section, we take inspiration from the IOC approach of [24] to define the loss function ℓ used to train the neural network previously introduced. Namely, we assume that the demonstrations $\{\tilde{\boldsymbol{\tau}}^{(d)}\}_{d=1}^{D}$ are optimal, i.e., they are optimal solutions to the QP problem (3) and thus satisfy its KKT conditions. As the QP constraints are satisfied during optimal demonstrations, the KKT conditions (*ii*)-(*v*) are automatically fulfilled. Therefore, determining the optimal parameters $\boldsymbol{\theta}^*$ of our neural network can be understood as searching for the parameters $\boldsymbol{\theta}$ fulfilling the first KKT condition for all the demonstrations. This corresponds to minimizing the loss

$$\ell(\boldsymbol{\tau}(\boldsymbol{\theta}), \{\tilde{\boldsymbol{\tau}}^{(d)}\}_{d=1}^{D}) = \sum_{d=1}^{D} \ell^{(d)}(\boldsymbol{\theta}) \tag{6}$$
$$\text{with} \quad \ell^{(d)}(\boldsymbol{\theta}) = \sum_{s} \|\nabla_{\boldsymbol{z}} \mathcal{L}(s, \boldsymbol{\theta}, \boldsymbol{z}, \tilde{\boldsymbol{z}}^{(d)}, \boldsymbol{\lambda}^{(d)})\|^2,$$

where we sum over the demonstrations and the progress of the task via the phase variable s. The Lagrangian of the problem (3) and its derivative for the d-th demonstration are

$$\begin{aligned} \mathcal{L}(s, \boldsymbol{\theta}, \boldsymbol{z}, \tilde{\boldsymbol{z}}^{(d)}, \boldsymbol{\lambda}^{(d)}) =& \frac{1}{2} \big(\tilde{\boldsymbol{z}}_s^{(d)} - \boldsymbol{z}_s\big)^{\mathsf{T}} \boldsymbol{W}_s(\boldsymbol{\theta}) \big(\tilde{\boldsymbol{z}}_s^{(d)} - \boldsymbol{z}_s\big) \\ &+ \boldsymbol{\lambda}_s^{(d)\mathsf{T}} \left(\boldsymbol{P}_s \boldsymbol{z}_s - \boldsymbol{r}_s\right), \\ \nabla_{\boldsymbol{z}} \mathcal{L}(s, \boldsymbol{z}, \boldsymbol{\theta}, \tilde{\boldsymbol{z}}^{(d)}, \boldsymbol{\lambda}^{(d)}) =& \boldsymbol{W}_s(\boldsymbol{\theta}) \big(\tilde{\boldsymbol{z}}_s^{(d)} - \boldsymbol{z}_s\big) + \boldsymbol{P}_s^{\mathsf{T}} \boldsymbol{\lambda}_s^{(d)}, \end{aligned}$$

where $\boldsymbol{z}_s \equiv \boldsymbol{z}(s)$, $\tilde{\boldsymbol{z}}^{(d)} = (\,\tilde{\boldsymbol{\xi}}_1^{(d)\mathsf{T}} \ldots \tilde{\boldsymbol{\xi}}_K^{(d)\mathsf{T}}\,)^{\mathsf{T}}$ is the vector of demonstrated skills outputs, $\boldsymbol{P} = \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{G} \end{pmatrix}$ and $\boldsymbol{r} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{h} \end{pmatrix}$ are the stacked constraints parameters, and $\boldsymbol{\lambda} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\nu} \end{pmatrix}$ is the vector of Lagrangian multipliers. Moreover, we can express $\boldsymbol{\lambda}^{(d)}$ in function of $\boldsymbol{\theta}$ for each demonstration d by minimizing the loss $\ell^{(d)}$ subject to the KKT complementary condition, i.e., $\nabla_{\boldsymbol{\lambda}^{(d)}} \ell^{(d)}(\boldsymbol{\theta}, \boldsymbol{\lambda}^{(d)}) = 0$. Therefore, by setting the optimization variable $\boldsymbol{z}_s$ to the output $\boldsymbol{z}_s^*(\boldsymbol{\theta})$ of our network, the loss of each demonstration is [3]

$$\ell^{(d)}(\boldsymbol{\theta}) = \sum_{s} \left\| \Big(\boldsymbol{I} - \boldsymbol{P}_s^{\mathsf{T}} \big(\boldsymbol{P}_s \boldsymbol{P}_s^{\mathsf{T}}\big)^{-1} \boldsymbol{P}_s \Big) \boldsymbol{W}_s(\boldsymbol{\theta}) \big(\tilde{\boldsymbol{z}}_s^{(d)} - \boldsymbol{z}_s^*(\boldsymbol{\theta})\big) \right\|^2.$$

The loss (6) inherently includes the task specifications via the demonstrations and the QP KKT conditions, and does not require additional task-specific design. To avoid the singular

[3] Equivalently, $\ell^{(d)}(\boldsymbol{\theta}) = \sum_s \|\boldsymbol{W}_s(\boldsymbol{\theta})(\tilde{\boldsymbol{z}}_s^{(d)} - \boldsymbol{z}_s^*(\boldsymbol{\theta}))\|^2$ for constant $\boldsymbol{P}_s$.

solution $\boldsymbol{W}_s(\boldsymbol{\theta}) = \mathbf{0}\ \forall s$, we leverage the softmax activation function, as explained next. Thus, at least one skill is given a high relative importance at each instant of the task.

D. Skills weights as positive-semidefinite matrices

As mentioned previously, the QP parameters are determined by the first part of our neural network. Specifically, the cost parameters are $\boldsymbol{Q}_s(\boldsymbol{\theta}) = \boldsymbol{W}_s(\boldsymbol{\theta})$, $\boldsymbol{c}_s(\boldsymbol{\theta}) = -\boldsymbol{W}_s(\boldsymbol{\theta})\hat{\boldsymbol{z}}_s$ where the weight matrix $\boldsymbol{W}_s(\boldsymbol{\theta})$ is learned by the network. The constraints parameters relate to skills outputs and to the robot physical characteristics. To obtain valid QPs, or equivalently to prevent skills to have negative relative importance weights, the weight matrices must be PSD, i.e., $\boldsymbol{W} \in \mathcal{S}_+^n$. We here describe two approaches to learn PSD weight matrices.

a) Diagonal weight matrices: In this case, we define

$$\boldsymbol{W}(\boldsymbol{\theta}) = \text{diag}\left(w_1(\boldsymbol{\theta})\boldsymbol{I}_1, \quad \dots \quad , w_K(\boldsymbol{\theta})\boldsymbol{I}_K\right), \tag{7}$$

where each block $w_k(\boldsymbol{\theta})\boldsymbol{I}_k$ weights the output of the k-th skill, and the scalars $\{w_k(\boldsymbol{\theta})\}_{k=1}^K$ are obtained from the fully-connected layer followed by a softmax activation function. The latter ensures that the scalar weights are positive and sum to 1, thus guaranteeing that $\boldsymbol{W}$ is PSD, and that at least one skill is activated at any instant of the task. Notice that we defined the different blocks as proportional to identity matrices to avoid altering the outputs of individual skills.

b) Full weight matrices: Such matrices allow us to express correlations between different skills, i.e, between their control values $\hat{\boldsymbol{\xi}}$, throughout the task. This naturally occurs in various tasks. For example, when approaching and grasping an object, the hand closure is correlated with the velocity at which the object is approached. We learn matrices

$$\boldsymbol{W}(\boldsymbol{\theta}) = \begin{pmatrix} w_1(\boldsymbol{\theta})\boldsymbol{I}_1 & \boldsymbol{W}_{12}(\boldsymbol{\theta}) & \dots & \boldsymbol{W}_{1K}(\boldsymbol{\theta}) \\ \boldsymbol{W}_{12}^\mathsf{T}(\boldsymbol{\theta}) & w_2(\boldsymbol{\theta})\boldsymbol{I}_2 & \dots & \boldsymbol{W}_{2K}(\boldsymbol{\theta}) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{W}_{1K}^\mathsf{T}(\boldsymbol{\theta}) & \boldsymbol{W}_{2K}^\mathsf{T}(\boldsymbol{\theta}) & \dots & w_K(\boldsymbol{\theta})\boldsymbol{I}_K \end{pmatrix}, \tag{8}$$

where the off-diagonal blocks $\boldsymbol{W}_{jk}$ encode the correlations between the outputs of the skills j and k. To guarantee the positive semidefiniteness of the matrices $\boldsymbol{W}$, we propose to learn the diagonal and off-diagonal blocks separately. Firstly, the scalar terms $\{w_k(\boldsymbol{\theta})\}_{k=1}^K$ are obtained as described in the previous paragraph. Secondly, the off-diagonal matrices $\{\boldsymbol{W}_{jk}(\boldsymbol{\theta})\}_{j,k=1}^K$ are obtained by leveraging the properties of matrices with positive block-diagonal elements [29], namely

$$\begin{pmatrix} \boldsymbol{Y} & \boldsymbol{X} \\ \boldsymbol{X}^\mathsf{T} & \boldsymbol{Z} \end{pmatrix} \in \mathcal{S}_+^n \iff \boldsymbol{X} = \boldsymbol{Y}^{1/2}\boldsymbol{K}\boldsymbol{Z}^{1/2}, \tag{9}$$

where $\boldsymbol{K}$ is a contraction matrix, i.e., $\|\boldsymbol{K}\| \leq 1$.Therefore, we use a second fully-connected layer to learn the contraction matrices as $\boldsymbol{K}_k = v_k(\boldsymbol{\theta})\frac{\boldsymbol{U}_k(\boldsymbol{\theta})}{\|\boldsymbol{U}_k(\boldsymbol{\theta})\|}$, with a tanh and a sigmoid activation function applied to $\boldsymbol{U}_k$ and v_k, respectively, so that $v_k \in [0, 1]$. The off-diagonal elements $\{\boldsymbol{W}_{jk}(\boldsymbol{\theta})\}_{j,k=1}^K$ are then computed recursively using the right-hand side of (9). For instance, in the case of a matrix composed of 3 skills, we first compute $\boldsymbol{X} = \boldsymbol{W}_{12}$ with $\boldsymbol{Y} = w_1\boldsymbol{I}_1$ and $\boldsymbol{Z} = w_2\boldsymbol{I}_2$, and then $\boldsymbol{X} = (\,\boldsymbol{W}_{13}^\mathsf{T}\ \boldsymbol{W}_{23}^\mathsf{T}\,)^\mathsf{T}$ with $\boldsymbol{Y} = \begin{pmatrix} w_1\boldsymbol{I}_1 & \boldsymbol{W}_{12} \\ \boldsymbol{W}_{12}^\mathsf{T} & w_2\boldsymbol{I}_2 \end{pmatrix}$ and $\boldsymbol{Z} = w_3\boldsymbol{I}_3$. Note that, to facilitate the training of full weight matrices, we initialize the parameters $\boldsymbol{\theta}$ of the scalar terms $\{w_k(\boldsymbol{\theta})\}_{k=1}^K$ with a previously-trained diagonal model.

V. Experiments

In this section, we evaluate our approach with different robotic platforms and manipulation tasks. All computations were performed on a laptop with 2.60GHz $\times 12$ CPU and 31 GiB RAM. A video of the experiments accompanies the paper (`https://youtu.be/00NXvTpL-YU`), and source codes are available at `https://github.com/NoemieJaquier/sequencing-blending/`.

A. Illustrative example: pick-and-place with planar robots

We first consider the pick-and-place task introduced in § IV-A and train our approach using diagonal and full weight matrices on the provided single manually-designed demonstration. In order to guarantee that one arm and one hand skill are activated at each instant of the task, we use one softmax activation function for each of the arm and gripper pairs of skills, namely pick/place and open/close. The task is then reproduced by the 4-DoF robot. As a baseline, we consider the case where the QP (4) with diagonal weights does not require additional constraints, so that its solution is $\dot{\boldsymbol{p}}^* = w_{\mathsf{pick}}\hat{\dot{\boldsymbol{p}}}_{\mathsf{pick}} + w_{\mathsf{place}}\hat{\dot{\boldsymbol{p}}}_{\mathsf{place}}$, $\dot{\boldsymbol{\gamma}}^* = w_{\mathsf{open}}\hat{\dot{\boldsymbol{\gamma}}}_{\mathsf{open}} + w_{\mathsf{close}}\hat{\dot{\boldsymbol{\gamma}}}_{\mathsf{close}}$. In this case, as the QP solution is readily available, we do not need to solve a nested optimization to minimize a given loss. Instead, the loss (6) can be minimized independently for each value of s with classical optimization methods. Finally, a 10-DoF student robot is requested to reproduce the learned sequence of skills with different pick and place positions. To do so, the pick and place DS skills are adapted to the new target points. For all reproductions, the QP is solved with respect to the arm and gripper joint velocities using (5).

Fig. 1b depicts the demonstrated trajectory, as well as the reproduction of the task by the 4-DoF robot with a diagonal weight matrix. Our approach successfully sequences the available skills and reproduces the task by picking and placing the object at the required locations. The differences of trajectory between the demonstration and the reproduction are due to the fact that the DS arm skills naturally follow a different trajectory than the demonstration between the target points (remember that the demonstration was generated independently from the given skills). For the same reason, the learned weights slightly differ from the manually-designed ones used to generate the demonstration (Fig. 1f). The differences of trajectory are attenuated when using a full weight matrix (see Fig. 1c), where correlations between skills are exploited to better match the demonstration. Note that only the diagonal weights are represented in Fig. 1f. As expected, the baseline looks similar to our approach with diagonal weight matrix (see Fig. 1d). Slight differences may be due to the different optimizations and to local minima in the loss. However, notice that the baseline applies only to very simple QPs, which are unrealistic for most applications (incl. for the experiments of § V-B- V-C). Also, in contrast to our approach, the baseline does not learn the weight matrix as a parametric function of the phase variable. Fig. 1e depicts the reproduction of the learned sequence by the 10-DoF robot using a diagonal weight matrix, showing that our approach successfully generalizes to different pick and place

locations. As the full weight matrix naturally overfits a single demonstration, it is not well suited to generalize in this case.

B. Pouring task with a humanoid robot

Here, we apply our approach in a real-world scenario to learn a complex sequence of skills on the humanoid robot ARMAR-6 [30]. The robot is positioned in front of a table, on which are placed an empty glass and a 1-liter plastic bottle partially filled with orange juice. The scenario consists of a pouring task, where the robot grasps the bottle, pours juice into the glass, and places the bottle back on the table. The positions of the objects are assumed a priori known by the robot, but could equally be inferred by a perception system.

As for the previous experiment, a set of skills is provided as black-box solutions. Specifically, four skills are defined for the arm, namely approach the bottle, pour, place the bottle back, and retreat the arm. Moreover, two joint-velocity-based skills are provided for the five-fingered hand, namely open and close in a power cylindrical grasp. The four arm skills $\{\mathsf{C}_{\mathsf{approach}}, \mathsf{C}_{\mathsf{pour}}, \mathsf{C}_{\mathsf{place}}, \mathsf{C}_{\mathsf{retreat}}\}$ are defined by DS with radial vector fields pointing toward a fixed point attractor. Their desired control values correspond to the end-effector linear and angular velocities $\hat{\dot{\boldsymbol{p}}}$ and $\hat{\dot{\boldsymbol{q}}}$, which depend on the current end-effector position $\boldsymbol{p}_s \in \mathbb{R}^3$ and orientation $\boldsymbol{q}_s \in \mathcal{S}^3$, i.e., $\hat{\boldsymbol{\xi}}(\psi_s) \equiv \begin{pmatrix} \hat{\dot{\boldsymbol{p}}}(\boldsymbol{p}_s) \\ \hat{\dot{\boldsymbol{q}}}(\boldsymbol{q}_s) \end{pmatrix}$. The fixed point attractors of the four arm skills are the robot hand grasp pose on the bottle for the approach skill, a tilted hand pose above the glass for the pour skill, the hand pose at the position of the bottle on the table for the place skill, and the hand resting pose for the retreat skill. The hand skills $\{\mathsf{C}_{\mathsf{open}}, \mathsf{C}_{\mathsf{close}}\}$ are defined similar to the gripper skills of the pick-and-place example, and thus open and close all finger joints by controlling their velocity. We train our approach on seven manually-designed demonstrations for which an operator defined the arm and hand trajectories. The bottle and glass positions were varied of ± 10 and ± 20 cm along the x and y axes, respectively. As previously, we use two softmax activation functions for the arm and hand skills, and the phase variable is $s = t/T$.

After the learning phase, the robot successfully reproduced the pouring task using both diagonal and full weight matrices (see Fig. 3 (*top-left*)). Moreover, our approach not only succeeded at learning the desired sequence of skills, but also resulted in seamless transitions as indicated by the absence of pauses and by the smoothness of the trajectories depicted in Fig. 3 (*bottom-left*). The learned weight matrices are represented in Fig. 3 (*right*) for the diagonal and full cases. Although the resulting trajectories look similar, the matrices still differ in the relative importance attributed to each skill. Notably, the model with full weight matrices exploits the correlation between the skills to shape the reproduced trajectory, thus featuring lower diagonal values than the diagonal model. Therefore, full weight matrices have better representation capabilities than their diagonal counterpart. However, this comes at the expense of generalization abilities. Indeed, as shown in Fig. 3, the diagonal model was able to generalize to bottle and glass locations that were outside the demonstrated range (here, the bottle and glass positions were swapped along the x axis), which the full model could only achieve for locations close to the demonstrations. Finally, we compared our approach to a baseline obtained by manually sequencing the given skills without any learning or blending. As shown in Fig. 3 (*bottom-left*), the baseline trajectory is characterized by obvious jerky transitions. The resulting timing would cause the robot to overfill the glass, thus failing the reproduction. Importantly, our approach is well-suited for learning and executing the sequence of skills on a real robot. Indeed, the pouring task training lasted a couple of minutes, and the testing time was $\sim$ 3-4 ms per timestamp, which allowed us to execute our approach at a control frequency of 200 Hz.

C. Bimanual sweeping task learned from human data

We aim at evaluating our approach to sequence and blend skills based on human demonstrations, i.e., on data for which no ground truth is easily available. To do so, we consider a bimanual sweeping task from the KIT motion database [31], [32], in which a human transfers cucumber slices from a cutting board to a bowl. At the beginning of the demonstrations, a subject stands in front of a table. A cutting board on which cucumber slices are placed, is positioned along the edge of the table in front of the human. The human first grasps a plastic bowl with the left hand and a knife with the right hand using cylindrical power grasps. Then, s/he holds the bowl below the table next to the cutting board, and pushes the cucumber slices into the bowl with the knife. Finally, the human places both knife and bowl back.

For the bimanual sweeping task, we consider the motion of each arm separately. Moreover, we use demonstrations of the aforementioned sweeping task performed by two different subjects. First, three naturally-varying demonstrations of the first subject are used to obtain a skill library. Here, we consider a set of four low-level skills per arm, namely $\{\mathsf{C}^{\mathsf{l}}_{\mathsf{approach}}, \mathsf{C}_{\mathsf{hold}}, \mathsf{C}^{\mathsf{l}}_{\mathsf{place}}, \mathsf{C}^{\mathsf{l}}_{\mathsf{retreat}}\}$ and $\{\mathsf{C}^{\mathsf{r}}_{\mathsf{approach}}, \mathsf{C}_{\mathsf{sweep}}, \mathsf{C}^{\mathsf{r}}_{\mathsf{place}}, \mathsf{C}^{\mathsf{r}}_{\mathsf{retreat}}\}$ for the left and right arm, respectively. Each human demonstration is manually segmented into four parts corresponding to the approach, sweep/hold, place, and retreat skills. In this experiment, we use via-points movement primitives (VMP) [27], which offer powerful skill representations that are easily adaptable to new starts, goals and via-points after training. Therefore, each skill is then represented by a time-dependent VMP trained on the corresponding segments of the demonstrations. The desired control values are the end-effector position and unit-quaternion-based orientation $\binom{\boldsymbol{p}}{\boldsymbol{q}}$ given by the mean trajectory retrieved by the VMPs. The desired control values depend on the time t_s, i.e., $\hat{\boldsymbol{\xi}}(\psi_s) \equiv \begin{pmatrix} \hat{\boldsymbol{p}}(t_s) \\ \hat{\boldsymbol{q}}(t_s) \end{pmatrix}$. All VMPs are executed with the start and goal poses defined by the desired task. The timing of the VMP skills is defined by the duration of the entire task T. Within our model, every skill trajectory is then evaluated at the evolving time $t = sT$ based on the overall phase variable s. The resulting skills are illustrated by Fig. 4a. As for the previous experiments, these skills are considered as black-box solutions, meaning that their representation is not directly known by our model. We then use three demonstrations provided by a second

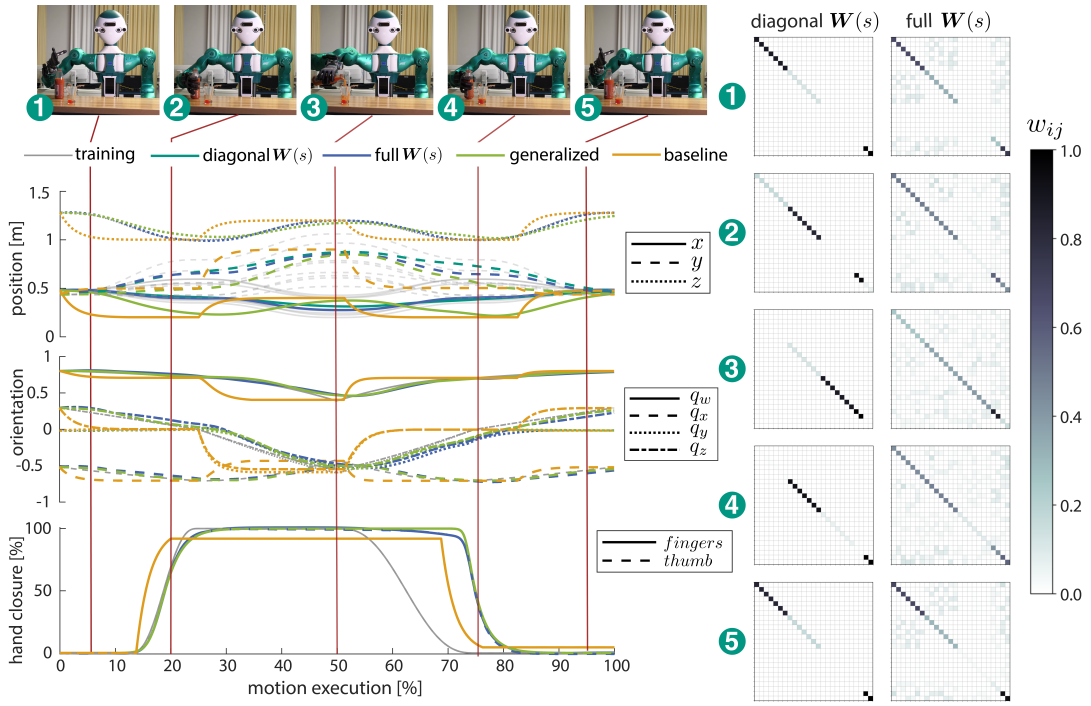
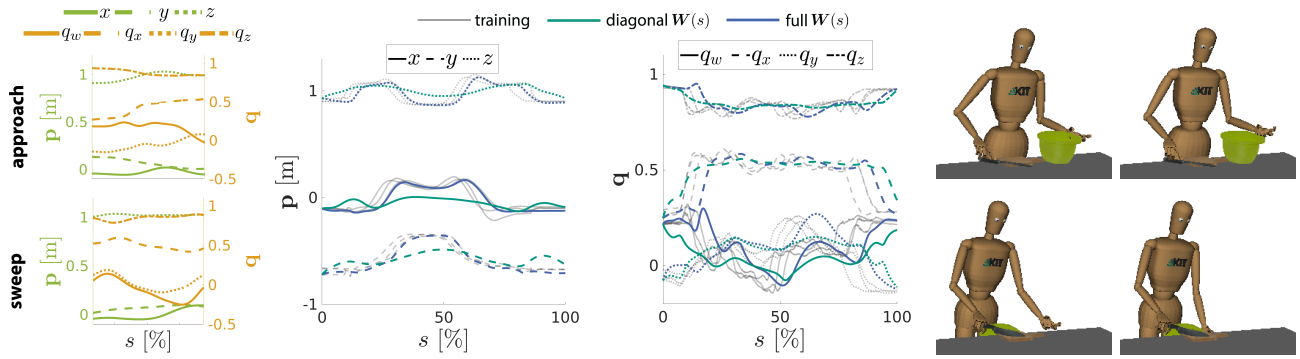


Fig. 3: Pouring task with a humanoid robot. The *top* row shows snapshots of the robot in the resting position (*1*) and executing the approach (*2*), pour (*3*), place (*4*) and retreat (*5*) skills during the task. The *bottom-left* graphs depict the demonstrated (—) and reproduced hand position, orientation, and closure trajectories. Reproductions are obtained with our approach using diagonal (—) and full (—) weight matrices. A generalized motion obtained with diagonal weight matrices (—), as well as a baseline where skills are manually sequenced without blending (—), are also displayed. The *right* column depicts the learned diagonal and full weight matrices at different task instants.



(a) $C_{approach}$, C_{sweep} (b) Position (left arm) (c) Orientation (right arm) (d) Diag. $W(s)$ (e) Full $W(s)$

Fig. 4: Bimanual sweeping task with a human model. *(a)* Approach and sweep VMP skills. *(b)-(c)* Demonstrations (—) and reproductions of the task using diagonal (—) and full (—) weight matrices. *(d)-(e)* Snapshots of the reproduction at $s = 0.25$ (*top*) and $s = 0.5$ (*bottom*).

different subject to train two models of our approach (left and right arm separately) with diagonal and full weight matrices. Note that these demonstrations include variations, as humans motions naturally vary across executions of the same task.

A simulated kinematic human model, as well as models of the bowl, knife and table, are used for the reproduction phase. In this case, the model with diagonal weight matrices could not reproduce the task as it was not able to closely fit the demonstrations (see Fig. 4b- 4e). This is due to the significant differences between the low-level skill trajectories (trained on the first subject) and the demonstrations (provided by the second subject). Notice that such differences also appeared in the pick-and-place experiment. However, as opposed to the sweeping task, the arm trajectories between the pick and place locations did not influence the task success, allowing both diagonal and full weight matrices to be used. For the bimanual sweeping task, only full weight matrices lead to a successful reproduction by learning correlations between skills. Notice that, although two separated models were trained for the left and right arms, the learned full weight functions conserved the timing of the motions, allowing both arms to be synchronized during the reproduction. Also, the training and testing times were similar to the pouring task.

VI. Conclusion

We proposed a skill-agnostic formulation to learn to sequence and blend skills using QP-based differentiable optimization layers. This allows us to represent the relative importance of skills as a function of the task progress and to

optimize it for a given loss with gradient-based approaches. Our experiments showed that, provided a set of black-box skills and one or few demonstrations of a task, our approach not only learns unknown sequences composed of various types of skills, but also generates smooth motions with seamless, blended transitions. Overall, our diagonal model is advantageous for generalization, while full weight matrices are beneficial when demonstrations must be closely followed.

It is worth noticing that the considered pouring and sweeping tasks are generally difficult to learn with a single model. Instead, our approach decomposes a task by combining several skills, which are easy to train and potentially re-usable across tasks. Moreover, it requires only one or few demonstrations of the complete task, making it less cumbersome to train than trial-and-error-based models. This is a major advantage compared to black-box optimization techniques used in multitask control, although detailed performance comparisons are deferred to future work. Finally, in contrast to end-to-end methods, our formulation is modular, fast to train, and interpretable as the relative importance of skills is directly embedded in the weight matrices.

Importantly, the performance of our approach highly depends on the capabilities of the given individual skills. Namely, a given task can be reproduced only if the provided skill library contains a set of skills that can be sequenced and combined to do so. Also, our approach generalizes to new object locations under the condition that the corresponding skills successfully adapt to these locations. The dependency of the model parameters to a time-driven phase variable also limits the generalization. This can be overcome by defining the phase variable as a time-independent, perception-based measure of task progress, which we will explore in the future.

One drawback of our approach is that the dimensionality of the optimization variable increases rapidly with the number of different types of skills, i.e., which provide different control variables. To be applied to cases featuring a complex library with many different types of skills, we will extend our approach to handle hierarchies of skills. For instance, high-level skills, e.g., sweeping cucumber slices to a bowl, may first be learned with our approach as sequences of low-level skills, and then combined in a complex task, e.g., preparing a salad, with an additional QP-based formulation. We will then evaluate our approach in more complex scenarios including, e.g., hierarchies, and soft prioritization of skills.

REFERENCES

[1] F. Mussa-Ivaldi and E. Bizzi, "Motor learning through the combination of primitives," *Philos. Trans. R. Soc. Lond., B, Biol. Sci.*, vol. 355, pp. 1755–1769, 2000.

[2] T. Flash and B. Hochner, "Motor primitives in vertebrates and invertebrates," *Curr. Opin. Neurobiol.*, vol. 15, no. 6, pp. 660–666, 2005.

[3] R. S. Johansson and J. R. Flanagan, "Coding and use of tactile signals from the fingertips in object manipulation tasks," *Nat. Rev. Neurosci.*, vol. 10, no. 5, pp. 345–359, 2009.

[4] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[5] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *ICML*, 2017.

[6] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *NeurIPS*, 2019.

[7] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations," *Rob. Auton. Syst.*, vol. 74, pp. 97–107, 2015.

[8] ——, "Probabilistic progress prediction and sequencing of concurrent movement primitives," in *IEEE/RSJ IROS*, 2015, pp. 449–455.

[9] L. Rozo, M. Guo, A. G. Kupcsik, M. Todescato, P. Schillinger, M. Giftthaler, M. Ochs, M. Spies, N. Waniek, P. Kesper, and M. Bürger, "Learning and sequencing of object-centric manipulation skills for industrial tasks," in *IEEE/RSJ IROS*, 2020, pp. 9072–9079.

[10] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *IJRR*, vol. 31, no. 3, pp. 360–375, 2012.

[11] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement Learning with Sequences of Motion Primitives for Robust Manipulation," *IEEE T-RO*, vol. 28, no. 6, pp. 1360–1370, 2012.

[12] M. Saveriano, F. Franzel, and D. Lee, "Merging position and orientation motion primitives," in *IEEE ICRA*, 2019, pp. 7041–7047.

[13] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Auton. Robot.*, vol. 42, no. 3, pp. 529–551, 2018.

[14] T. Luksch, M. Gienger, M. Mühlig, and T. Yoshiike, "Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems," in *IEEE/RSJ IROS*, 2012, pp. 2082–2088.

[15] M. Mühlig, A. Hayashi, M. Gienger, S. Iba, and T. Yoshiike, "Receding horizon optimization of robot motions generated by hierarchical movement primitives," in *IEEE/RSJ IROS*, 2014, pp. 129–135.

[16] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions," in *IEEE ICRA*, 2011, pp. 1283–1290.

[17] N. Dehio, R. F. Reinhart, and J. J. Steil, "Multiple task optimization with a mixture of controllers for motion generation," in *IEEE/RSJ IROS*, 2015, pp. 6416–6421.

[18] V. Modugno, U. Chervet, G. Oriolo, and S. Ivaldi, "Learning soft task priorities for safe control of humanoid robots with constrained stochastic optimization," in *IEEE/RAS Humanoids*, 2016, pp. 101–108.

[19] Y. Su, Y. Wang, and A. Kheddar, "Sample-efficient learning of soft task priorities through Bayesian optimization," in *IEEE/RAS Humanoids*, 2018, pp. 1–6.

[20] J. Li, Y. Zhu, L. Huo, and Y. Chen, "Sample-efficient learning of soft priorities for safe control with constrained Bayesian optimization," in *IEEE IRC*, 2020, pp. 406–407.

[21] K. Bouyarmane and A. Kheddar, "On weight-prioritized multitask control of humanoid robots," *IEEE Trans. Autom. Control*, vol. 63, no. 6, pp. 1542–1557, 2016.

[22] C. Collette, A. Micaelli, C. Andriot, and P. Lemerle, "Robust balance optimization control of humanoid robots with multiple non coplanar grasps and frictional contacts," in *IEEE ICRA*, 2008, pp. 3187–3193.

[23] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Berkeley Symp. on Mathematical Statistics and Probability*, 1951, pp. 481–492.

[24] P. Englert, N. A. Vien, and M. Toussaint, "Inverse KKT: Learning cost functions of manipulation tasks from demonstrations," *IJRR*, vol. 36, no. 13-14, pp. 1474–1488, 2017.

[25] M. Pirotta and M. Restelli, "Inverse reinforcement learning through policy gradient minimization," in *AAAI*, 2016, pp. 1993–1999.

[26] E. Gribovskaya, S. M. Khansari-Zadeh, and A. Billard, "Learning nonlinear multivariate dynamics of motion in robotic manipulators," *IJRR*, vol. 30, no. 1, pp. 80–117, 2011.

[27] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter- and extrapolation capabilities," in *IEEE/RSJ IROS*, 2019, pp. 4301–4308.

[28] S. M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Rob. Auton. Syst.*, vol. 62, no. 6, pp. 752–765, 2014.

[29] R. Bhatia, *Positive Definite Matrices*. Princeton University Press, 2007.

[30] T. Asfour, M. Wächter, L. Kaul, S. Rader, P. Weiner, S. Ottenhaus, R. Grimm, Y. Zhou, M. Grotz, and F. Paus, "ARMAR-6: A high-performance humanoid for human-robot collaboration in real world scenarios," *IEEE RAM*, vol. 26, no. 4, pp. 108–121, 2019.

[31] C. Mandery, O. Terlemez, M. Do, N. Vahrenkamp, and T. Asfour, "Unifying representations and large-scale whole-body motion databases for studying human motion," *IEEE T-RO*, vol. 32, no. 4, pp. 796–809, 2016.

[32] F. Krebs, A. Meixner, I. Patzer, and T. Asfour, "The KIT bimanual manipulation dataset," in *IEEE/RAS Humanoids*, 2020-2021.