2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

# Synthesizing Compliant Reaching Movements by Searching a Database of Example Trajectories

Miha Deniša[1], Tadej Petrič[1], Tamim Asfour[2], and Aleš Ude[1,*]

*Abstract*— **We address the problem of generating new compliant reaching movements by searching a structured database of example trajectories. The proposed control framework is a multi-step process, where in the first step a human tutor teaches the robot how to perform a set of example reaching movements. In the second step, the recorded motion trajectories are executed with different velocities using a high gain feedback controller, for the purpose of learning corresponding torque control signals. The commanded torques are measured and stored together with the trajectory data. This data is organized in a hierarchical, graph-like structure, thereby providing the basis for search for new compliant trajectories, which can consist of parts of the previously acquired example movements. The proposed approach can construct a complete representation for newly discovered movements, including the feedforward torque commands. Finally, in the last step, the motion is executed using a low gain feedback controller and the associated feedforward torque signal. This ensures sufficient tracking accuracy and at the same time compliant behavior, which allows smooth interaction with the environment and is safe for cooperative task execution with humans. The usefulness of the proposed method was shown on a Kuka LWR robot.**

## I. INTRODUCTION

A well established approach for dynamic robot control is the use of inverse dynamic models [1]. However, due to the increasing complexity of robot mechanisms such as humanoid robots, the accurate dynamical models are often difficult to obtain. To fulfil the gap, algorithms for machine learning were adopted in robotics because of their ability of learning complex models. Although learning algorithms became powerful enough to learn even the inverse dynamics [2], they still require a large amount of data for learning. As an alternative, different biology inspired methods were proposed for dynamic robot control. An extensive review of computational mechanisms for sensorimotor control, which covers methods from optimal feedback control [3] to the forward models and predictive control [4], was recently published by Franklin and Wolpert [5].

Inspired by the human sensorimotor ability, which can learn arbitrary dynamic tasks, we propose a new control framework that is based on learning a task dependent trajectory with corresponding control signals. The proposed framework is a multi step process, where in the first step, human tutor teaches the robot how to perform the desired task, e. g. a reaching movement. In the second step, the corresponding task dependent control signals are learned by executing the movement using a high gain feedback control loop, which ensures sufficient tracking accuracy. In the last step, the desired reaching movement is executed using the feedforward task dependent control signal and low gain feedback loop, which ensures compliance and stability. Because of the feedforward compensation, the robot will perform the desired task with similar accuracy as in the second step. Due to the low feedback gain, the robot exhibits a compliant behavior with low perturbation rejection, therefore it is safer for humans to work with.

However, just building a database of movements with the associated control signals might not be an optimal solution. It is not feasible to obtain all the necessary movement trajectories and their task dependent control signals for the entire workspace based on user demonstrations only. In this paper we propose to augment the set of available movement primitives by a hierarchical database search. A set of movement trajectories, which partly share a similar course of movement, can be used to discover new movements. Such an approach can significantly reduce the teaching effort, since a database that contains various example motions with similar sections can be used to generate new, not previously demonstrated movement trajectories. Through the hierarchical database search, new behaviors can be discovered, generated and eventually added to the database.

The proposed hierarchical search for new movement primitives is based on the work done mainly in the computer graphics community, which has long studied how to utilize large databases of diverse movements. This is in contrast to most of the work done in robotics, which is primarily focused on learning from a single demonstration [6] or learning from multiple demonstrated variations of the same type of movement [7]. It was shown that by organizing movements in motion graphs, smooth transitions between interconnected full-body movements can be found [8]. This approach was applied by Kovar et al. [9] to generate different styles of locomotion along arbitrary paths. In robotics, a graph-based representation similar to motion graphs was used by Yamane et al. [10]. They combined transition graphs with a binary tree database in order to generate human body locomotions. This research was later expanded to plan object receiving motions [11].

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience, and from the Slovenian Research Agency under grant agreement no. J2-4284.

[1] M. Deniša, T. Petrič, and A. Ude are with the Jožef Stefan Institute, Department of Automatics, Biocybernetics and Robotics, Humanoid and Cognitive Robotics Lab, Ljubljana, Slovenia miha.denisa@ijs.si, tadej.petric@ijs.si, ales.ude@ijs.si

[2] T. Asfour is with the Karlsruhe Institute of Technology, Institute for Anthropomatics, High Performance Humanoid Technologies Lab, Karlsruhe, Germany asfour@kit.de

Besides in Yamane et al. [10], movement generation from trajectory libraries has also been investigated in [12], [13], [14], [15]. Like in our work, graph search is utilized in [12], but this work is focused on locomotion. Ude et al. [13] used variations of the same primitive movement to generate a new instantiation of a dynamic movement primitive that is optimal for a given situation. Like in [13], a situation descriptor is used also in [14] to transfer previously optimized trajectories to new situations. Mülling et al. [15] studied the problem of mixing dynamic movement primitives to generate optimal striking movements. The work described in this paper is most closely related to [10], [11] because our focus is on how to generate new dynamic movement primitives from parts of previously acquired example trajectories.

The rest of the paper is organized as follows. In Section II an approach for compensating robot dynamics is presented. It consists of Section II-A describing task trajectory learning, Section II-B describing learning of torque control signals, and Section II-C that deals with the learned trajectory execution. Section III proposes an approach for discovering new movement primitives thorugh hierarchical graph search. This is accomplished by building a database (Section III-A) and then using it to find and synthesize new compliant movements (Section III-B). The paper concludes with experimental results and conclusions.

## II. COMPENSATION OF ROBOT DYNAMICS

To compensate for robot dynamics, we normally apply a generic approach which is based on inverse dynamical model of the robot. Assuming that the robot consists of rigid bodies, the joint space equations of motion are given by

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\varepsilon}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{1}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the joint positions, velocities and accelerations, respectively, $\mathbf{H}(\boldsymbol{q})$ is the inertia matrix, $\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ are the Coriolis and centripetal forces, $\boldsymbol{g}(\boldsymbol{q})$ are the gravity forces and $\boldsymbol{\varepsilon}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ are the nonlinearities not considered in the rigid body dynamics, e. g. friction. We denote the inverse dynamic model of the robot (1) as $\mathbf{f}_{\text{dynamic}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$. Using the inverse dynamic model, a possible control approach for tracking the desired joint positions $\boldsymbol{q}_d$ is given by

$$\boldsymbol{\tau}_{cmd} = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\dot{\boldsymbol{q}}) + \mathbf{f}_{\text{dynamic}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}), \tag{2}$$

where $\mathbf{K}$ is the diagonal matrix for stiffness and $\mathbf{D}(\dot{\boldsymbol{q}})$ is the damping term. Note that high values in matrix $\mathbf{K}$ stiffen the robot, which results in better tracking and error rejection in case of perturbations. On the other hand, if the stiffness values are low, the robot is compliant but the tracking accuracy might be poor.

To have both advantageous properties, i. e. accurate tracking and compliance, we proposed a new multi-step control system which includes feedforward torque signal that corresponds to the desired trajectory. Essentially, it is a pre-generated internal model-based control system. However, instead of using a complete inverse dynamic model for compensating the robot dynamics as usually, we use a set of layers based on dynamic movement primitives, which encode the information of the torque control signals alongside with the desired path (motion trajectories). The main advantage of the proposed control system is that is model free, i. e. the dynamic model of the robot is not needed. Moreover, since torque signal that corresponds to the desired task are feedforward during the execution step, the high tracking accuracy and natural compliant behavior are achieved. Natural compliance is the compliance of the mechanism itself. The proposed control system ensures that the robot is always compliant during the execution of the task, thereby ensuring that the collision contact forces are small and therefore the robot can perform tasks in unstructured environment and safely interact with humans.

### A. Learning task trajectories

In the first step, the goal is to learn the motion trajectories (positions) demonstrated by a human teacher. Different techniques exist for teaching a desired motion to the robot; one can use kinesthetic guiding [16], haptic interfaces [17], motion capture systems [18], [19], etc. Kinesthetic guiding was used in this paper.

To encode motion trajectories, we use Dynamic Movement Primitives (DMPs). They are summarized in [6]. The equations below are valid for one degree of freedom (DOF). For multiple DOFs the equations can be used in parallel. For one DOF they are defined by the following nonlinear system of differential equations

$$\tau_{dmp}\dot{v} = \alpha_z(\beta_z(g-y)-v) + f(x), \tag{3}$$

$$\tau_{dmp}\dot{y} = v. \tag{4}$$

The linear part of Eq. (3) – (4) ensures that y converges to the desired final configuration, here denoted as g. The nonlinear part $f(x)$ modifies the shape of the movement and is defined by a linear combination of radial basis functions

$$f(x) = \frac{\sum_{i=1}^{N} w_i \psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} \tag{5}$$

$$\psi_i(x) = \exp(-h_i(x-c_i)^2), \tag{6}$$

where $\psi_i$ defines the basis functions with centers at $c_i$ and widths $h_i > 0$. As seen in Eq. (5), $f(x)$ is not directly time dependent. Instead, phase variable $x$ defined in Eq. (7), with initial value $x(0) = 1$, is used to make the dependency more implicit:

$$\tau_{dmp}\dot{x} = -\alpha_x x \tag{7}$$

The phase is common across all DOFs. By specifying the time evolution through phase, it becomes easier to stop the clock in case of external perturbations, which cause the robot to deviate from the desired trajectory. It can be shown that – given the properly defined constants $\alpha_z$, $\beta_z$, $\tau_{dmp}$, $\alpha_x > 0$ – the above system is guaranteed to converge to the desired final configuration $g$.

We can encode demonstrated trajectories as DMPs by applying locally weighted regression and learn the target function defined as

$$f_j(t) = \tau_{dmp}\ddot{q}_j(t) + \tau_{dmp}\alpha_y\dot{q}_j(t) - \alpha_y\beta_y(g - q_j(t)), \tag{8}$$

where $q_j(t)$ denotes the demonstrated trajectory of the $j$-th joint at time $t$.

### B. Learning torque control signals

In the second step we encode corresponding control torque signals for the kinematic trajectory $\boldsymbol{q}_d$, which was learned in the first step. To obtain the corresponding torque signals, we employed a high gain feedback controller, which ensured required tracking accuracy. The feedback control is given by

$$\boldsymbol{\tau} = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\dot{\boldsymbol{q}}), \tag{9}$$

Since kinematic trajectory is time invariant and the corresponding control torque signals must be time dependent, we introduce a task time multiplier $\kappa$ that defines the duration of the task. With this in mind, DMP equations (3), (4), and (7) used for executing the demonstrated task trajectories while learning torque control signals, can be rewritten as:

$$\kappa\tau_{dmp}\dot{v} = \alpha_z(\beta_z(g-y)-v) + f(x), \tag{10}$$

$$\kappa\tau_{dmp}\dot{y} = v, \tag{11}$$

$$\kappa\tau_{dmp}\dot{x} = -\alpha_x x. \tag{12}$$

The equations for encoding and learning of the torque control signals are similar as given in section II-A. The main difference is that instead of learning the kinematic trajectory $q_d$, we learn the target function given by

$$f_j(t) = \kappa\tau_{dmp}\ddot{\tau}_j(t) + \kappa\tau_{dmp}\alpha_y\dot{\tau}_j(t) - \alpha_y\beta_y(g-\tau_j(t)) \tag{13}$$

where $\tau_j$ is the commanded torque signal for the $j$-th joint. By learning the control torque $\boldsymbol{\tau}_{ff}$, which is produced by the high gain feedback controller, the system essentially learns the corresponding inverse dynamics $\mathbf{f}_{\text{dynamic}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ along the executed kinematic trajectory $\boldsymbol{q}_d$.

### C. Executing the desired motion

In this step, the movement trajectory $\boldsymbol{q}_d$ and the corresponding torque control signal $\boldsymbol{\tau}_{ff}$ is executed, using a low gain feedback controller. The controller used for executing the motion is given by

$$\boldsymbol{\tau} = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\dot{\boldsymbol{q}}, C) + \boldsymbol{\tau}_{ff}, \tag{14}$$

where $\boldsymbol{\tau}_{ff}$ is the feedforward torque control signal, which was learned in the second step (note that feedforward torques are only active in the movement execution step). According to the control system analysis from [20], when feedforward models are used, a low-gain feedback loop is sufficient to preserve the stability of the system. On the other hand, without feedback loop the system would inevitably diverge, regardless of the precision of the feedforward model.

The main advantages of using feedforward models are the better tracking performance (compared to a system without feedforward terms) and the possibility to use low-gain feedback, which enables natural compliant behavior of the robot. Note that low-gain feedback has little impact on the mechanical (natural) compliance of the robot.

## III. NEW TASK TRAJECTORIES

In the previous section we described how to acquire a set of desired trajectories with the corresponding torque control signals associated with the movement executions at different speeds. At this point the recorded trajectories can be played back using feedforward torque control signals. In this section we investigate how to combine the available trajectories to generate new trajectories with the corresponding feedforward torque control signals using hierarchical graph search. The application of hierarchical graph search for the generation of new robot movements as such is not new, see e. g. [10]. Here we study how to add the corresponding feedforward torque control signals to the newly found trajectories.

### A. Building the database

To combine two different types of information in a hierarchical database, we use a similar approach as Yamane et al. [11]. Instead of storing and relating the motion of two subjects in an integrated database as in [11], we store and relate the kinematic trajectories $\boldsymbol{q}_d$ and the associated torque control signals $\boldsymbol{\tau}_{ff}$. The first part of the database thus represents the kinematic trajectories $\boldsymbol{q}_d$ obtained by kinesthetic guiding. We concatenate them in a sample position matrix:

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n], \tag{15}$$

where $\boldsymbol{x}_i$ denotes the state vectors sampled at a given discrete time interval and $n$ is the total number of all samples belonging to all learned kinematic trajectories incorporated into the database. State vectors are defined as

$$\boldsymbol{x}_i = [q_{1i}, \dot{q}_{1i}, q_{2i}, \dot{q}_{2i}, \ldots, q_{di}, \dot{q}_{di}]^T, \tag{16}$$

where $j$-th joint angle and its velocity at time $t_i$ are denoted by $q_{ji}$ and $\dot{q}_{ji}$, respectively, and $d$ is the number of the robot degrees of freedom.

We use the sample joint matrix as a root node of a binary tree, which represents learned position trajectories. We use $k$-means algorithm (with $k = 2$) to cluster similar state vectors and thus split the root node into two child nodes. The data in each of these nodes is then clustered again to gain the nodes at the next level of the binary tree, as shown in Fig. 1.

Criterion for when to stop splitting the tree nodes is based on the variability of data contained in the node. We define the mean distance $d_k$ of node $k$ as

$$d_k = \frac{\sum_{i=1}^{n_k} d(\boldsymbol{x}_{ki}, \boldsymbol{c}_k)}{n_k}, \tag{17}$$

where $n_k$ denotes the number of state vectors clustered at node $k$. $d(\boldsymbol{x}_{ki}, \boldsymbol{c}_k)$ is the Euclidean distance between state vectors $\boldsymbol{x}_{ki}$ associated with node $k$ and the node's centroid $\boldsymbol{c}_k$, which was calculated by the $k$-means algorithm. If $d_k$ is lower than a predefined threshold, then state vectors contained in the node are similar and we stop splitting this node. With this we avoid the binary tree getting unnecessarily deep while ensuring the needed precision of the representation. With this criterion we cluster the data into nodes until we do not have any nodes left to split. To ensure that all state vectors are

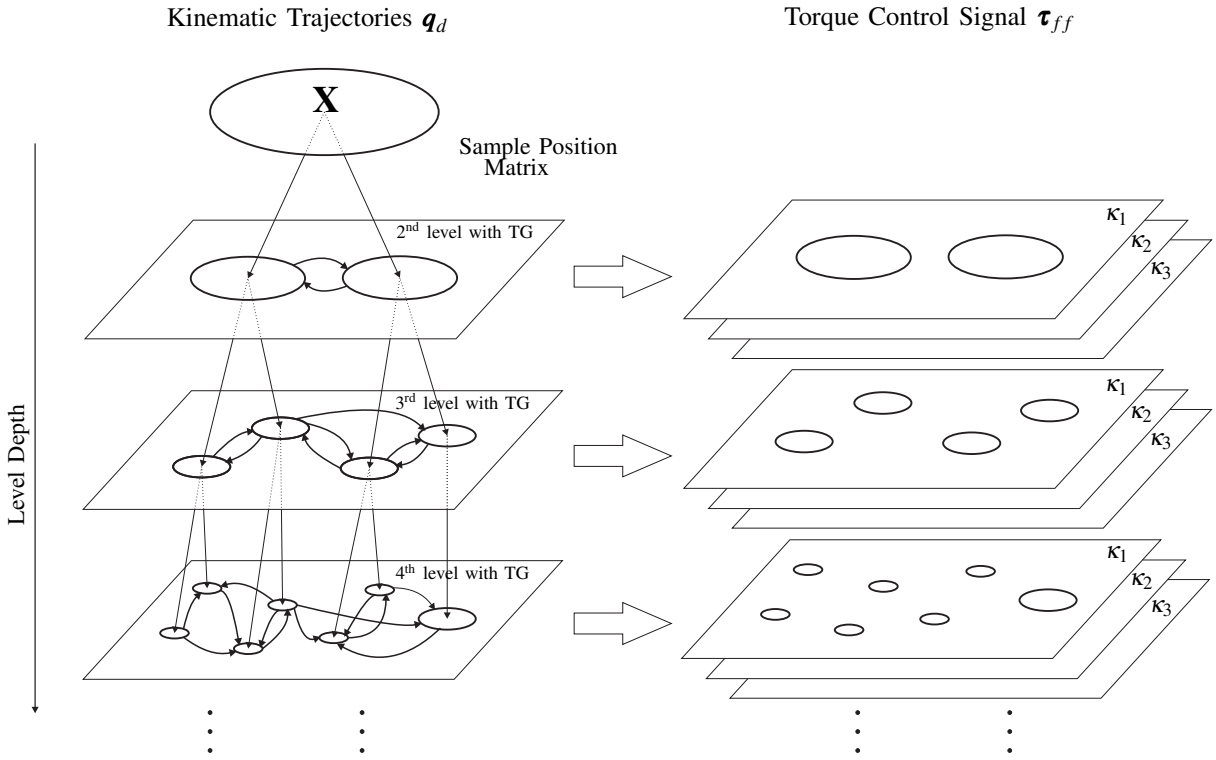


Fig. 1. Both parts of the database. The figure shows its structure, with kinematic trajectories represented on the left side and corresponding torque trajectories on the right. The sample joint matrix $\mathbf{X}$ is divided into two child nodes with $k$-means clustering. Then, the transition graph (TG), which represents probabilistic transitions between the nodes at this level, is built. The data associated with each node is clustered into child nodes for the 3rd level, where the TG is build again. We continue this procedure until all nodes fit the stopping criteria. Note that we expand those nodes to the last level and thus represent all of the data at all levels. The right side represents corresponding torques at different speed of execution i. e. different task time multipliers $\kappa$. At every level of the database, each node in the binary tree, on the left, has one or more (example figure shows three) corresponding torque means and time durations, on the right.

represented at all levels of the binary tree, every branch is extended to the last level.

Transition graph, representing all possible transitions between the nodes, is built at each level of the tree (see Fig. 1). The edge weights in the transition graph represent the probability of transition from one node to another. Transition probability from node $k$ to node $l$ is estimated by

$$P_{kl} = \frac{m_{kl}}{n_k} \tag{18}$$

where $m_{kl}$ denotes the number of transitions observed in all trajectories of the original data, i. e. the number of all state vectors clustered in node $k$ that have a successor in node $l$.

For further processing it is not necessary to store all state vectors $\boldsymbol{x}_k$ at each node of the binary tree. Instead, only the mean of the corresponding state vectors $\overline{\boldsymbol{x}}_k$ is stored at each node. If the node contains exactly one start/final configuration, we store it instead of the mean. In this way we ensure that movements generated by graph search end in the same end points as the learned position trajectories. In this step the time component is lost. We explain in Section III-B how time duration is estimated.

As we are synthesizing new task trajectories consisting of kinematic trajectories $\boldsymbol{q}_d$ and control torques $\boldsymbol{\tau}_{ff}$, the database must also encode torque control signals. We do not separately cluster the torque signals, but rather associate the torques with the corresponding nodes in the transition graph. This means that for each part of the kinematic trajectories, represented in a single node through the mean of state vectors, we store the corresponding means of torque signals $\overline{\boldsymbol{\tau}}_{ff}$ and time durations $t_d$. We do this multiple times as we execute the same movements with different time duration, i. e. with different task time multipliers $\kappa$. See Fig. 1 for example representation of the whole database.

### *B. Searching for new task trajectories*

We start the search for new trajectories by selecting the desired start and end points on two different trajectories. In addition, the desired task time multiplier $\kappa$ is selected. A binary tree level also needs to be selected. As the level determines the fidelity of reproduction compared to the original trajectories, we normally select the last level of the binary tree. We then try to find a path between the nodes corresponding to the desired start and end joint position. To achieve that we employ A* search algorithm in the transition graph at the desired level. As long as the two trajectories share a similar enough part, the most probable path is found and with it a sequence of nodes i. e. mean state vectors.

Based on the selected time multiplier $\kappa$, we add the corresponding mean torques $\overline{\boldsymbol{\tau}}_{ff}$ to the state vectors $\overline{\boldsymbol{x}}_k$ of the

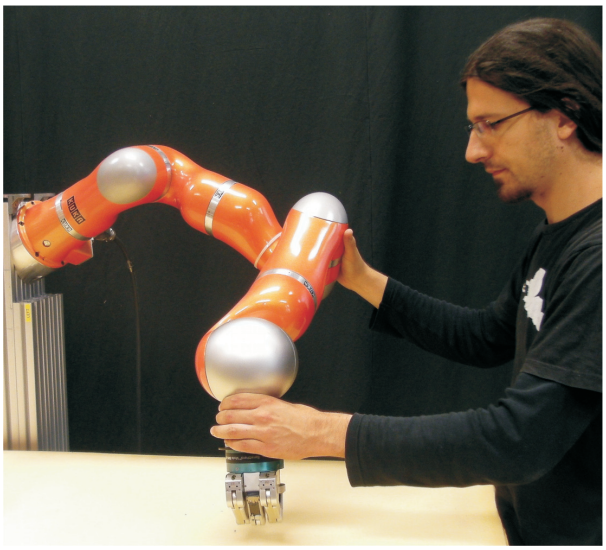

Fig. 2. Learning task trajectories by kinesthetically guiding the Kuka LWR robot arm.

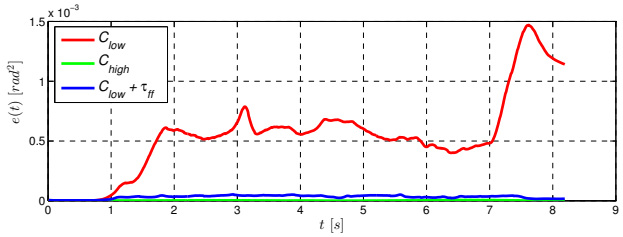


Fig. 3. The sum of all joint's tracking errors as defined in (20). The green line represents tracking errors while executing the movement with a high gain feedback controller ($C_{high}$). Red line represents tracing error while using a low gain feedback controller ($C_{low}$). Finally, the blue line represents the sum of all joint's tracking errors while using the proposed controller, low gain feedback controller and feedforward torque signals ($C_{low}+\tau_{ff}$). We can observe low tracking errors gained by incorporating feed-forward torque signals. Note that the gain used for feed-back loop controller $C_{low}$ was 20 times lower than $C_{high}$.

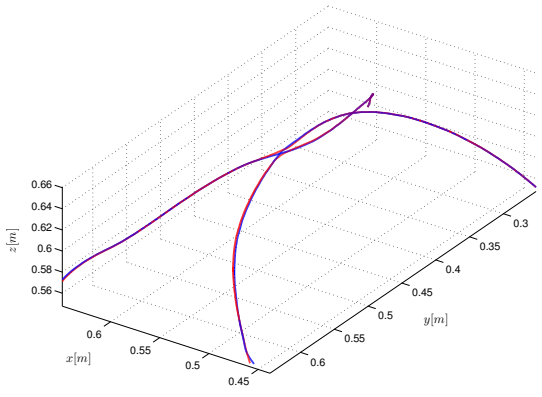


Fig. 4. The closest demonstrated and newly synthesized kinematic trajectories in task space. Blue lines represent demonstrated trajectories, while red lines represent newly synthesized kienmatic trajectories.

discovered sequence. We enhance this sequence further with time durations $t_d$ corresponding to the added torques. Now we have a sequence of joint positions, torques and their time durations. The newly discovered sequence can be written as:

$$\left\{(\bar{\boldsymbol{x}}_1,\bar{\boldsymbol{\tau}}_{ff1},0),\left(\bar{\boldsymbol{x}}_2,\bar{\boldsymbol{\tau}}_{ff2},\frac{t_{d1}+t_{d2}}{2}\right),\ldots\right.$$
$$\left.\ldots,\left(\bar{\boldsymbol{x}}_{n_p},\bar{\boldsymbol{\tau}}_{ffn_p},\frac{t_{d(n_p-1)}+t_{dn_p}}{2}\right)\right\}, \quad (19)$$

where $n_p$ denotes the number of nodes on the trajectory.

We synthesize a trajectory from each discovered sequence by encoding it as a DMP. The DMPs describing newly synthesized kinematic trajectories and the corresponding torque control signals (also encoded as DMPs) can then be used to execute new, not directly shown, movements while remaining compliant, as described in Section II-C.

## IV. EVALUATION

We evaluated the proposed approach using a Kuka LWR robot arm. The demonstrator taught the robot several reaching movements while kinesthetically guiding the arm (see Fig. 2). Two of the learned movements that intersect each other are shown in Fig. 4. All kinematic trajectories were encoded as DMPs. They were used in the second step to obtain corresponding torque control signals, as described in Section II-B. Each movement was executed three times with different task time multipliers $\kappa=\{1,2,3\}$. The learned movement trajectories $\mathbf{q}_d$ and the corresponding torque control signals $\boldsymbol{\tau}_{ff}$ were then used to execute the learned reaching movements using a low-gain feedback controller (14). Fig. 3 shows an example sum of all joint's tracking errors with respect to time

$$e(t)=\sum_{j=1}^{7}(q_{aj}(t)-q_j(t))^2, \quad (20)$$

where $q_{aj}(t)$ denotes the actual $j$-th joint position. We can observe that by using a low-gain feedback loop without the feed forward torque signal $\boldsymbol{\tau}_{ff}$ in order to achieve compliance, tracking error escalates greatly (red line) in comparison to a high-gain feedback loop (green line). However, by adding the feedforwad torque control, similar compliance can be achieved, while successfully tracking the desired trajectory $\boldsymbol{q}_d$ (blue line). Note that the low gain was 20 times lower than during the execution with a high-gain feedback loop.

Both the learned kinematic trajectories and the corresponding torque signals were used to build the database, as described in Section III-A. This database was used to find new reaching movements as described in section II-B. A* search algorithm found new sequences of nodes, as the demonstrated trajectories had parts that were sufficiently similar. Each new sequence started in the first node of one of the demonstrated trajectories and ended in the final node of one of the others. Each new sequence of mean position $\bar{\boldsymbol{x}}$ was then enhanced with mean torques $\bar{\boldsymbol{\tau}}_{ff}$ and corresponding time duration $t_d$ three times, once per task time multiplier $\kappa$. Using DMPs we found complete representations of new

TABLE I

MEAN TRACKING ERRORS AND STANDARD DEVIATIONS FOR EACH JOINT AND THEIR SUMS WHEN EXECUTING REACHING MOVEMENTS. $Task_1$ AND $Task_2$ DENOTE TWO EXAMPLES OF THE DEMONSTRATED MOVEMENTS, WHILE $Task_{12}$ AND $Task_{21}$ DENOTE TWO EXAMPLES OF NEWLY SYNTHESIZED REACHING MOVEMENTS. ALL VALUES ARE IN $[10^{-5} rad^2]$.

| $Task_1$ ($\kappa = 1$) | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $\Sigma$ |
|---|---|---|---|---|---|---|---|---|
| $C_{high}$ | 0.306 (0.289) | 1.18 (1.69) | 0.660 (0.491) | 1.89 (1.65) | 0.696 (1.23) | 0.211 (0.292) | 0.693 (1.09) | 5.64 (6.74) |
| $C_{low}$ | 112 (101) | 69.9 (77.7) | 48.7 (28.3) | 41.9 (98.0) | 27.5 (44.9) | 56.9 (145) | 7.29 (11.6) | 364 (507) |
| $C_{low} + \tau_{ff}$ | 2.51 (1.96) | 21.7 (24.9) | 1.54 (1.25) | 5.49 (6.67) | 3.31 (4.90) | 1.79 (2.61) | 5.55 (9.32) | 41.9 (51.7) |

| $Task_2$ ($\kappa = 1$) | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $\Sigma$ |
|---|---|---|---|---|---|---|---|---|
| $C_{high}$ | 0.325 (0.235) | 0.617 (0.938) | 0.930 (0.989) | 0.200 (0.292) | 0.191 (0.197) | 0.0836 (0.143) | 0.405 (0.552) | 2.75 (3.35) |
| $C_{low}$ | 100 (74.8) | 48.5 (75.4) | 31.3 (25.7) | 38.1 (47.8) | 9.41 (14.2) | 12.3 (22.0) | 4.94 (8.46) | 245 (268) |
| $C_{low} + \tau_{ff}$ | 2.77 (2.93) | 21.7 (26.5) | 2.23 (2.53) | 4.12 (7.43) | 1.58 (1.90) | 2.29 (3.38) | 2.40 (3.47) | 37.1 (48.2) |

| $Task_{12}$ $C_{low} + \tau_{ff}$ | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $\Sigma$ |
|---|---|---|---|---|---|---|---|---|
| $\kappa = 1$ | 29.3 (44.4) | 18.6 (22.6) | 14.0 (24.6) | 7.12 (9.87) | 9.32 (12.5) | 2.98 (3.26) | 1.31 (2.25) | 82.7 (119) |
| $\kappa = 2$ | 27.2 (61.5) | 8.88 (7.04) | 3.40 (9.56) | 8.01 (12.2) | 15.9 (21.8) | 4.28 (4.89) | 2.07 (2.21) | 69.8 (119) |
| $\kappa = 3$ | 26.7 (68.4) | 17.0 (15.7) | 3.76 (7.74) | 6.49 (8.97) | 17.8 (24.6) | 3.02 (3.71) | 1.23 (1.66) | 76.0 (131) |

| $Task_{21}$ $C_{low} + \tau_{ff}$ | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ | $q_6$ | $q_7$ | $\Sigma$ |
|---|---|---|---|---|---|---|---|---|
| $\kappa = 1$ | 7.29 (6.42) | 26.4 (19.9) | 3.62 (3.30) | 7.24 (9.59) | 6.82 (11.5) | 4.53 (4.60) | 1.11 (2.52) | 57.0 (57.8) |
| $\kappa = 2$ | 3.63 (3.86) | 10.2 (8.01) | 1.02 (1.80) | 8.97 (13.7) | 8.62 (11.77) | 8.90 (3.60) | 1.74 (1.67) | 43.1 (44.3) |
| $\kappa = 3$ | 2.16 (3.07) | 21.1 (17.0) | 1.74 (1.93) | 6.97 (11.5) | 8.67 (11.8) | 5.86 (4.51) | 1.21 (1.48) | 47.7 (51.3) |

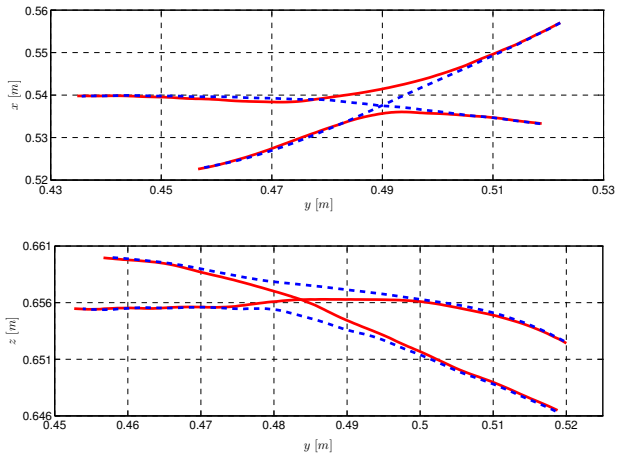


Fig. 5. Sections of the closest demonstrated trajectories and the newly synthesized trajectories in the task space. These sections represent transitions of new trajectories, represented with red lines, from one demonstrated trajectory to the other, represented with dashed blue lines. We can observe smooth and continuous transitions.

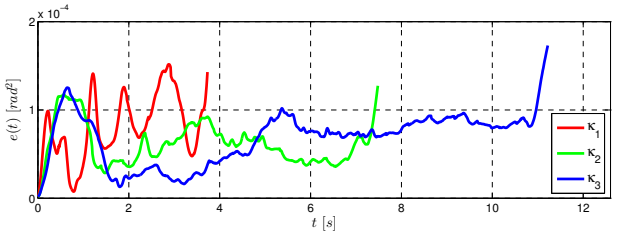


Fig. 6. The sum of all joint's tracking errors (20) of newly synthesized tasks. Errors of three example tasks are given. They were executed with different task time multipliers $\kappa$. Even though these movements were never directly shown, the tracking error remains within a tolerable range.

reaching movements trajectories. Two new example position trajectories in task space can be seen in Fig. 4, marked with red lines. For clarity, sections of demonstrated and new kinematic trajectories in two different 2D spaces are shown in Fig. 5.

All new reaching movements were executed on the robot as proposed in Section II-C. Fig. 6 shows joint position tracking errors of three example movements with different task time multipliers. Tracking errors are again defined as in Eq. 20. Tracking errors of newly synthesized tasks remain similar to those of directly demonstrated tasks.

Table I contains means and standard deviations of tracking errors $(q_{aj}(t) - q_j(t))^2$ for individual joints $j$. All values are in $10^{-5} rad^2$. First two parts of the table ($Task_1$ and $Task_2$) show errors with respect to the execution of the two closest reaching movements. They were both executed with task time multiplier $\kappa = 1$. Errors for this part of the table were measured by performing reaching movements with high gain feedback controller ($C_{high}$), low gain feedback controller ($C_{low}$), and low gain feedback controller with feedforward torque signal ($C_{low} + \tau_{ff}$). Low gain used for all movements was 20 times lower than high gain. We can observe how the tracking error drastically increases when we apply low gain instead of high gain. But there is a significant drop in

error when we add feedforward torque signals to low gain feedback control. The last two parts of the table contain errors measured when executing two examples of the newly synthesized reaching movements ($Task_{12}$ and $Task_{21}$) while using low gain feedback control with feedforward torque signals. Each movement was executed for all three task time multipliers $\kappa$. We can immediately observe the consistency of tracking error over different multipliers $\kappa$. Secondly, there is no significant increase in tracking error even though these movements were newly synthesized. We can thus conclude that newly synthesized reaching movements can be executed using low gain feedback control with feedforward torque signals.

## V. CONCLUSIONS

We proposed and evaluated an approach to discover new reaching trajectories in a database of example trajectories and to learn the corresponding dynamics. By feedforwarding the associated torque control signals, we can execute the reaching movements with a high tracking accuracy while exhibiting compliant behavior without using a full dynamic model. We showed that new reaching movements can be generated from a library of kinesthetically guided example movements, as long as they have sufficiently similar partial trajectories. In our experiments we used the developed approach to acquire a library of several reaching movements. Each of the kinesthetically guided movements was executed at different velocities using high gain controller and the associated torque control signals were stored in the database together with the kinematic trajectory. New movements were found using graph search. Our evaluation showed that newly synthesized movements maintain the needed tracking accuracy and compliance even though they were built from parts belonging to different movements. With a sufficient number of example movements, we could execute any reaching movement with compliant behavior while maintaining accuracy, without the need for a full dynamical robot model.

## REFERENCES


[1] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer, 2000.

[2] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey." *Cognitive Processing*, vol. 12, no. 4, pp. 319–40, 2011.

[3] C. G. Atkeson and J. M. Hollerbach, "Kinematic features of unrestrained vertical arm movements." *The Journal of Neuroscience*, vol. 5, no. 9, pp. 2318–30, 1985.

[4] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control." *Neural Networks*, vol. 11, no. 7-8, pp. 1317–29, 1998.

[5] D. W. Franklin and D. M. Wolpert, "Computational mechanisms of sensorimotor control." *Neuron*, vol. 72, no. 3, pp. 425–42, Nov. 2011.

[6] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[7] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian Mixture Models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[8] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and adverbs: multidimensional motion interpolation," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32–40, 1998.

[9] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics*, vol. 21, no. 3, July 2002.

[10] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," *Autonomous Robots*, vol. 30, no. 1, pp. 87–98, 2010.

[11] K. Yamane, M. Revfi, and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013, pp. 1621–1628.

[12] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson, "Transfer of policies based on trajectory libraries," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, 2007, pp. 2981–2986.

[13] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[14] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, pp. 111–127, 2013.

[15] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, vol. 23, no. 3, pp. 263–279, 2013.

[16] D. Kushida, M. Nakamura, S. Goto, and N. Kyura, "Human direct teaching of industrial articulated robot arms based on force-free control," *Artificial Life and Robotics*, vol. 5, no. 1, pp. 26–32, 2001.

[17] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: object-lifting case study with a humanoid," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, 2009, pp. 399–404.

[18] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 93–108, June 2004.

[19] J. Babic, J. G. Hale, and E. Oztop, "Human sensorimotor learning for humanoid robot skill synthesis," *Adaptive Behavior*, vol. 19, no. 4, pp. 250–263, 2011.

[20] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. Wiley, 2008.