Chapter 1

A Review of Compliant Movement Primitives

Miha Deniša, Tadej Petrič, Andrej Gams and Aleš Ude

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/64058

Abstract

Dynamical models of robots performing tasks in contact with objects or the environment are difficult to obtain. Therefore, different methods of learning the dynamics of tasks have been proposed. In this chapter, we present a method that provides the joint torques needed to execute a task in a compliant and at the same time accurate manner. The presented method of compliant movement primitives (CMPs), which consists of the task kinematical and dynamical trajectories, goes beyond mere reproduction of previously learned motions. Using statistical generalization, the method allows to generate new, previously untrained trajectories. Furthermore, the use of transition graphs allows us to combine parts of previously learned motions and thus generate new ones. In the chapter, we provide a brief overview of this research topic in the literature, followed by an in-depth explanation of the compliant movement primitives framework, with details on both statistical generalization and transition graphs. An extensive experimental evaluation demonstrates the applicability and the usefulness of the approach.

Keywords: compliant movements, adaptive system, learning system, robot control, learning by demonstration

1. Introduction

The need to operate in unstructured environments, such as human everyday environments and homes, drives the development of algorithms for fast generation of new trajectories of robots in compliant behavior mode without sacrificing tracking accuracy. Operating in unstructured environments and with humans requires compliant robot behavior because of possible unplanned contact with objects, and more importantly, with humans themselves. To achieve such behavior, accurate dynamical models of the robot and the task are needed [1].

Yet, acquisition of accurate dynamical models of robots and more specifically of robots performing a task, are difficult to obtain. Therefore, different methods, including biologically inspired methods, were proposed for robot control [2]. Other approaches have been proposed for torque learning. For example, Nguyen-Tuong and Peters [3, 4] relied on local Gaussian process regression (GPR) and used it for on-line dynamic model learning. Their approaches improve the accuracy of the model while avoiding high feedback gains. On the other hand, their method requires the availability of a large quantity of data in order to fully learn a complete dynamic model, and not only task-specific torques. Learning of torques for a specific task can be utilized using iterative learning control (ILC) [5] as was shown in the paper of Schwarz and Behnke [6], who used ILC to learn motor and friction models. Similarly, Gautier et al. [7] proposed an iterative learning identification and control method for dynamic robot control.

The latest generations of robotic mechanisms, such as the Kuka LWR-4, are equipped with joint-torque sensors [8], which can be used to measure the torques during the operation. The possibility of recording joint torques has been exploited in several approaches for learning of task-specific joint torques. One such method is with the use of compliant movement primitives (CMPs), first introduced by Petrič et al. [9] and later adopted by [10, 11], which encode both the kinematic trajectory as well as the corresponding joint torques.

The main topic of this chapter is a review of compliant movement primitives (CMPs), which are suitable for robots with active torque control. CMPs enable accurate execution while maintaining a compliant mode of operation, without requiring explicit models of task dynamics. CMPs draw their inspiration from the human ability to learn arbitrary dynamical tasks [12]. They can be easily learned from user demonstrations. In this paper, we present the basics of the CMPs, followed by the means to surpass the limited applicability of pure imitation through generation of new, previously untrained movements. In order to do so, two mathematical means were exploited: statistical generalization, which allows for variation in task configuration, and hierarchical database search, which allows combinations of previously trained tasks.

Both statistical generalization and hierarchical database search have been previously employed in robotics on a kinematic level. For example, generation of kinematic trajectories with statistical generalization using locally weighted regression was shown in [13]. The method was applied to dynamic movement primitives (DMPs) [14], which also constitute the kinematic part of the CMPs. Similarly, Forte et al. used Gaussian process regression to generalize between the weights of the DMPs [15]. Other approaches of generalization, not relying on DMPs, are thoroughly discussed in [16]. Hierarchical database presented in this chapter consist of transition graphs, that is, motion graphs on each level. It was shown that smooth transitions between movements can be found if they are organized in motion graphs [17]. While Koval et al. [18] used motion graphs to generate different styles of locomotion, Yamane et al. [19, 20] combined them with a binary tree database. Similar work on hierarchical databases was also done by Deniša et al. [11, 21].

Compliant movement primitives are composed of both kinematic and dynamic trajectories. While the first, kinematic part, is encoded as the aforementioned DMPs, the latter, dynamic

part, is encoded as a set of radial basis functions. Only the combination of both allows accurate and compliant execution of trajectories. Similar approaches have recently appeared in the literature. An approach that utilizes tactile sensors to determine the force of contact with the environment on the iCub robot, and calculate the joint torques from the measured arm pose was proposed by Calandra et al. [22]. The authors propose using calculated joint-torques in a feed-forward manner for the control, just as is the case in CMPs. Similarly, Steinmetz et al. [23] recorded joint torques along the kinematic trajectory, encoded as a DMP, and used these torques as a feed-forward signal for the controller to increase the accuracy in the next execution of the in-contact task.

The need for robot operations in unstructured environments, combined with the complexity of the acquisition of dynamical models of various tasks, and the occurrence of similar methods in the literature all point at the applicability of the proposed compliant movement primitives approach for robots that go beyond the factory floor.

This chapter is structured as follows. We first provide the details the CMPs, explaining both the kinematic and the dynamic aspects. Generation of new trajectories with statistical generalization and hierarchical database search is explained in Section 3. Experimental evaluation is given in Section 4, followed by a Discussion and a Conclusion.

2. Compliant movement primitives

Compliant movement primitives CMPs are defined as a combination of Dynamic movement primitives (DMPs) for encoding kinematic trajectories and corresponding task-specific dynamics encoded in Torque Primitives (TPs). They are defined as

$$\boldsymbol{h}(t)=\left[\ \ddot{\boldsymbol{p}}_d(t),\ \dot{\boldsymbol{p}}_d(t),\ \boldsymbol{p}_d(t),\boldsymbol{\tau}_f(t)\right], \tag{1}$$

where $\ddot{\mathbf{p}}_d(t)$, $\dot{\mathbf{p}}_d(t)$, $\mathbf{p}_d(t)$, are the desired acceleration, velocity, and position encoded in the DMPs and $\boldsymbol{\tau}_f(t)$ are the corresponding task-specific dynamics, that is, joint torques or forces, encoded in TPs.

The learning of CMPs is done in three different stages. The first stage is learning the kinematic trajectory for the DMPs. The second stage is learning of corresponding torque or force profiles for TPs, and the last, third, stage is the execution of CMPs. While the learning of kinematic trajectories in the form of DMPs is well documented [13–15, 24], the literature for learning TPs is not as vast [10, 25].

2.1. Control structure

The basic control structure for robot control using CMPs is shown in **Figure 1**. The main advantage of the proposed control architecture is the model free approach which in the

execution step enables natural compliant behavior while ensuring sufficient accuracy of the desired movement. Natural compliance is the compliance of the mechanism itself. Because the robot is compliant while executing the task, the forces in case of an unforeseen collision are small. Thus, the robot can perform tasks in unstructured environment and safely interact with humans.

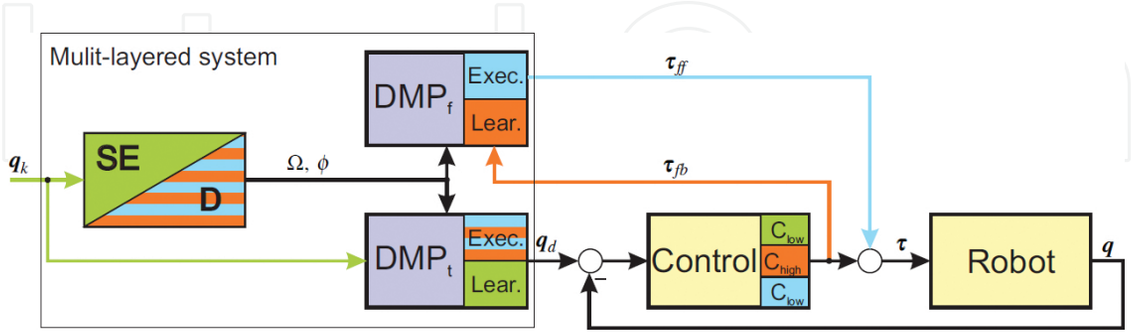

Figure 1. Block diagram of the multi-layered and multi-stage control system. The colored lines (green, orange, and blue) are only active when a chosen state is active (see text for detailed description).

Figure 1 shows the structure of the proposed multi-step control algorithm. The colors indicate which block is active in each step: green—learning of the DMPs, orange—learning of the TPs, and blue—execution of the CMPs. Note that black connections are always active, regardless of the step.

Assuming the robot consists of rigid bodies, the joint space equations of motion can be written in a form

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{g}(\boldsymbol{q})+\dot{\boldsymbol{o}}(\ddot{\boldsymbol{q}},\dot{\boldsymbol{q}},\boldsymbol{q})=\boldsymbol{\tau}, \tag{2}$$

where $\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}$ are the joint acceleration, velocity, and position, respectively; $\boldsymbol{H}(\mathbf{q})$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ are the Coriolis and centripetal forces, $\mathbf{g}(\mathbf{q})$ are the gravity forces, and $\epsilon(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q})$ are additional nonlinearities, for example, friction. We denote the full robot dynamic model as $\mathbf{f}_{robot}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q})$. Note that this inverse dynamic model $\mathbf{f}_{robot}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q})$ does not include dynamics of the task.

In general, the common approach for tracking the desired motion is using the impedance control given with

$$\boldsymbol{\tau}_u = \boldsymbol{K}\left(\boldsymbol{q}_d-\boldsymbol{q}\right)+\boldsymbol{D}\left(\dot{\boldsymbol{q}}_d-\dot{\boldsymbol{q}}\right)+\boldsymbol{f}_{robot}(\ddot{\boldsymbol{q}},\dot{\boldsymbol{q}},\boldsymbol{q})+\boldsymbol{\tau}_f, \tag{3}$$

where **K** and **D** are the diagonal matrices of the desired stiffness and damping respectively [1], $\boldsymbol{q}_d$ is the vector of desired motion encoded in DMPs, and $\boldsymbol{\tau}_f$ is the vector of task-specific torques encoded in TPs. Here, if the values of **K** are high, the robot is accurately tracking the trajectory with high error rejection ratio, that is, with stiff behavior. Vice versa, if the values of **K** are low,

the robot cannot accurately track the desired trajectory unless additional task-specific torques $\boldsymbol{\tau}_f$ are provided.

In the following, we explain the three steps of the CMPs learning approach: (1) learning of DMPs, (2) learning of TPs, C) execution of CMPs with accurate trajectory tracking and compliant behavior.

2.2. Learning of DMPs

The aim of the first step was to learn the task-specific trajectories of motion, encoded in DMPs. There are several possibilities on how to gain and encoded the motion in DMPs for both periodic and point-to-point motions [13–14, 24]. A short recap fallows on how to encode motions based on human demonstrations for point-to-point DMPs. The equations below are valid for one DOF and can be used in parallel for multiple DOFs. A DMPs is defined as a nonlinear system of differential equations

$$v\dot{z} = \alpha_z\left(\beta_z\left(g - y\right) - z\right) + f\left(s\right), \tag{4}$$

$$v\dot{y} = z. \tag{5}$$

Here, the linear part ensures that y converges to the desired goal configuration once $f(s)$ becomes zero. $f(s)$ is a nonlinear part that defines the shape of the movement. It is given by

$$f\left(s\right) = \frac{\Sigma_{b=1}^{L_d} w_{qb}\psi_b(s)}{\Sigma_{b=1}^{L_d}\psi_b(s)}. \tag{6}$$

Here, the Gaussian basis functions $\psi_b(s)$ are defined as

$$\psi_b\left(s\right) = exp\left(-d_b\left(s - c_b\right)^2\right), \tag{7}$$

where c_b are centers and d_b are widths of the Gaussians. Since $f(s)$ is not directly time dependent, the phase variable s was introduced as

$$v\dot{s} = -\alpha_s s. \tag{8}$$

The phase variable is common across all DMPs and TPs, for example, it is common for all CMPs. With proper selection of parameters α_z, β_z, α_s and v, the convergence of the system is guaranteed. For evaluation, the parameters were set empirically to $\alpha_y = 48$, $\beta_z = \alpha_z/4$ and $\alpha_s = 2$.

To acquire weights w_{qb} that represent the desired motion, the target for learning is derived from Eqs. (4) and (5). It is given by

$$f_{target} = v^2 \ddot{q}_{xn}(t_i) + \alpha_z \dot{q}_{xn}(t_i) - \alpha_z \beta_z \left(g - q_{xn}(t_i)\right), i = 1, \ldots, T, \tag{9}$$

where the goal g is defined by the end value of the example trajectory $q_{xn}(t_T)$. To calculate weights w_{qb}, the overdetermined system (Eq. (9)) is solved using regression technic for each joint. Note that recursive regression [26] is also possible, which is a common choice for online imitation learning.

2.3. Learning of TPs

Once DMPs are learned, learning of TPs follows. This step is denoted with orange color in the block scheme given in **Figure 1**. Here, we give a short recap of the method that exploits stiff and accurate robot behavior in a supervised environment.

The TPs are given as a combination of Gaussian functions denoted by

$$\tau_f = \frac{\Sigma_{b=1}^{L_d} w_{tb} \psi_b(s)}{\Sigma_{b=1}^{L_d} \psi_b(s)}. \tag{10}$$

To gain corresponding torques for accurate tracking of the trajectory, the gain $\boldsymbol{K}$ is set to "high." Note that this usually implies stiff robot behavior; therefore, the action must be performed under human supervision. To acquire weights w_{tb} that represent the corresponding torques, the target for learning is given in a form of

$$f_{TP-target} = K\left(q_d - q\right) + D\left(\dot{q}_d - \dot{q}\right). \tag{11}$$

where y is the desired motion trajectory encoded in DMPs (Eqs. (4) and (5)). Using the same approach as in DMPs, the weights w_{tb} from Eq. (10) are calculated using a recursive regression [26] for each joint. The details on learning TPs are in [10].

2.4. Execution of CMPs

Once both, the DMPs and TPs are learned, the CMPs can be executed using the control law given by Eq. (3). This step is denoted with blue color in the block scheme given in **Figure 1**. Because the task-specific dynamics is provided in a feed-forward manner, the tracking accuracy remains high even if the feedback gains are much lower that during learning. By selecting lower feedback gains, it is assumed that the robot behavior is compliant. However, since feedback gains are low and the robot behavior is complaint, the error rejection ratio is small, from which follows that the tracking is only accurate as long as the system is not heavily perturbed.

Note that the main idea of the CMPs is that the feed-forward part assures the nominal behavior of the robot for a specific given task even in cases when low feedback gains are used. In this way, good tracking and compliant behavior are achieved at the same time.

3. Autonomous generation of CMPs

Learning CMPs by human demonstration greatly simplifies execution of tasks in a compliant manner, as it does not need mathematically defined dynamic models in advance. But doing this for each variation of a dynamically versatile task is unpractical and time-consuming. In the following, two example expansions of the initial CMP database are presented to overcome this issue. While the first subsection tackles the issue of generating completely CMPs using a search in hierarchical databases with transition graphs, the second delivers the methodology for using statistical techniques in order to generalize CMPs.

3.1. Hierarchical database search

This section presents combining available trajectories to generate new trajectories with the corresponding feed-forward torque control signals using hierarchical graph search. Generating new robot movements through hierarchical graph search as such is not new, see for example, [20, 21]. To apply the hierarchical graph search on CMPs, the database of CMPs needs to be divided into two parts: the primary part that includes the kinematic trajectories and the secondary one that includes the dynamic part of CMPs, that is, corresponding torques. As new kinematic trajectories are synthesized through hierarchical search in the initial database, corresponding torques are extracted from the secondary part of the database.

3.1.1. Building the database

The primary part of the database, which stores kinematic part of CMPs, that is, position trajectories $\boldsymbol{q}_d$, is a binary tree-like structure with transition graphs at each level. As the primary database is built, the secondary part is added. It encodes the dynamic part of CMPs, that is, corresponding torques $\boldsymbol{\tau}_f$. Database construction begins with concatenating initial example position trajectories into a sample position matrix

$$\boldsymbol{X} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_F\right], \tag{12}$$

where $\boldsymbol{x}_i$ denotes a state vector sampled at a given discrete time interval, and F the total number of all samples belonging to all example trajectories included in the database. A state vector, defined as

$$\boldsymbol{x}_i = \left[q_{x1}, \dot{q}_{x1}, q_{x2}, \dot{q}_{x2}, \ldots, q_{xP}, \dot{q}_{xP}\right], \tag{13}$$

where subscript x denotes examples, and P number of DOF represents positions and velocities of an example trajectory at a given discrete time interval.

The sample motion matrix, encapsulating all example potion trajectories, represents a root node of a binary tree. See **Figure 2** for a simple representation of the database. A k-means algorithm, with $k = 2$, is used to cluster similar state vectors and split the initial root node into two child nodes. Clustering is then used again on each of the two child nodes, and thus, the nodes at the next level are gained. This can also be seen in **Figure 2** depicting a simple example of a database. The nodes keep splitting until the criterion, based on the variability of the data in node, is met. The mean distance d_k of a node k, used as a "stop split" criterion, is defined as

$$d_k = \frac{\Sigma_{i=1}^{n_k} d\left(\boldsymbol{x}_{ki}, \boldsymbol{c}_k\right)}{n_k}, \tag{14}$$

where n_k denotes the number of state vectors clustered in node k. Euclidian distance $d(\boldsymbol{x}_{ki}, \boldsymbol{c}_k)$ is calculated between each state vector in the node and the node's centroid $\boldsymbol{c}_k$, gained by the k-means algorithm. As this criterion indicates the similarity of the clustered state vectors, the database does not get unnecessary deep, while the precision of the representation remains satisfactory. The clustering is continued until all nodes meet the "stop split" criterion. Every branch is extended as a leaf node until the last layer. In this way, all the state vectors are represented at all the database levels.

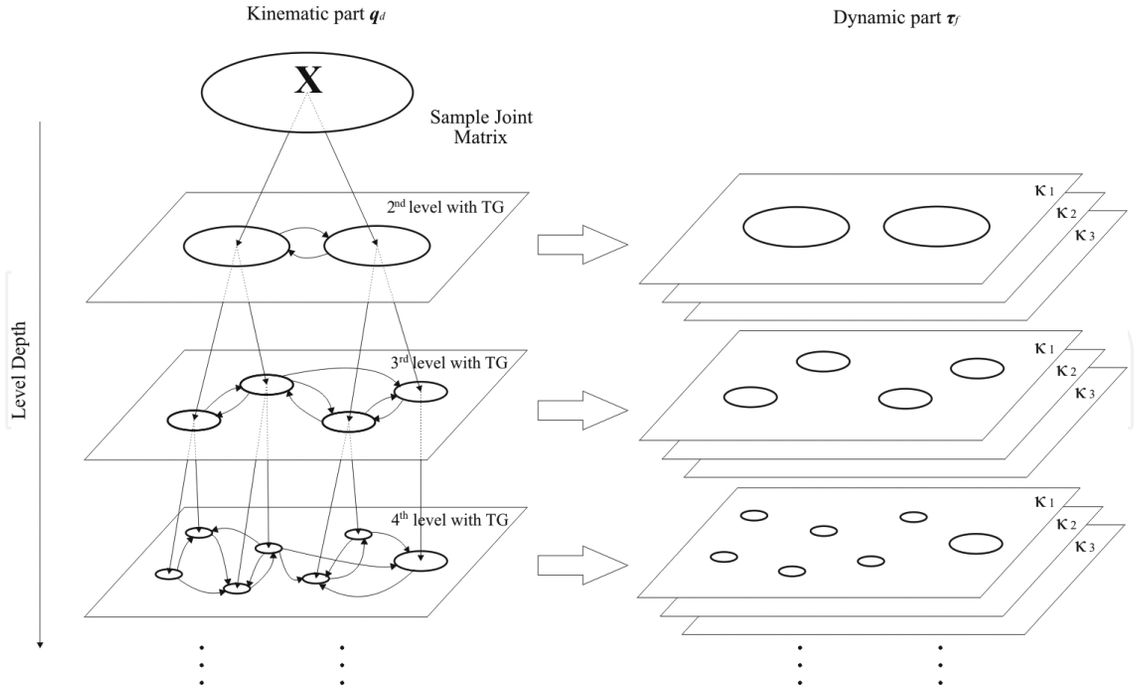

Figure 2. A simple representation of used database. The primary part encoding kinematic trajectories and with transition graphs is shown on the left. The secondary part which encapsulates corresponding torques and has multiple layers per level is depicted on the right side.

Every level of the database includes a transition graph representing all possible transitions between the nodes (see **Figure 2**). The graph's edge weights define the probability of transition from one node to the other. The transition probability from node k to node l is estimated by

$$P_{kl} = \frac{m_{kl}}{n_k} \tag{15}$$

where m_{kl} denotes the number of all state vectors clustered in node k that have a successor in node l, that is, the number of transitions observed in all trajectories of the original data. As state vectors are not needed any more, they are omitted, and at each node, a mean of corresponding state vectors $\bar{\mathbf{x}}_k$ is saved instead. At this point, the time component is also lost, which is tackled later on in this section.

While the already constructed primary part of database encodes CMP's kinematic trajectories, the secondary part will encode the dynamic part. The torques signals are not separately clustered, but rather associated with corresponding nodes in the primary database. For each part of the kinematic trajectories, represented in a single node through the mean of state vectors, a corresponding mean of the torques signal $\bar{\tau}_f$ and its time durations is stored. As the same kinematic movement can have different dynamic parts, this is done multiple times for each dynamic task descriptor. See **Figure 2** for example representation of the whole database.

3.1.2. Synthesizing new CMPs

A new trajectory is defined by first selecting the desired start and end points on two different trajectories. A dynamic task descriptor, for example, the desired task time multiplier k, is also selected. As the level determines, the fidelity of reproduction compared to the original trajectories, the last level of the hierarchical database is usually selected. A path between the nodes corresponding to the desired start and end joint position needs to be found. To achieve that A* search algorithm, it is used on 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, that is, mean state vectors.

Based on the selected dynamic task descriptor, for example, k, we add the corresponding mean torques $\bar{\boldsymbol{\tau}}_f$ to the state vectors $\bar{\mathbf{x}}_k$ of the discovered sequence. We enhance this sequence further with time durations t_d corresponding to the added torques. A sequence of positions, torques, and their time durations is gained. The newly discovered sequence can be written as

$$\left\{(\bar{\boldsymbol{x}}_1, \boldsymbol{\tau}_{f1}, 0), \left(\bar{\boldsymbol{x}}_2, \boldsymbol{\tau}_{f2}, \frac{t_{d1}+t_{d2}}{2}\right), \ldots, \left(\bar{\boldsymbol{x}}_r, \boldsymbol{\tau}_{fr}, \frac{t_{d(r-1)}+t_{dr}}{2}\right)\right\}, \tag{16}$$

where r denotes the number of nodes on the trajectory.

A trajectory from each discovered sequence is generated by encoding it as a DMP. The DMPs describing newly synthesized kinematic trajectories and the corresponding torque control signals (encoded as TPs) can then be used to execute new, not directly shown, movements while remaining compliant.

3.2. Statistical generalization

Using programming by demonstration approaches to learn CMPs can simplify compliant execution of dynamically versatile tasks. But executing demonstrations for each possible variation of the task would be time-consuming and cumbersome. For each new task descriptor, a new CMP needs to be learned, that is, motion trajectory needs to be learned through human demonstrations and executed on a robot for torque learning. Even if the deviation happens just on the torque level, for example, because of different payload, supervised learning of the torque is needed. Besides using actions graphs to generate new CMPs, as seen in the previous section, statistical generalization techniques can be employed. For that, a set of learned CMPs which transition smoothly between each other as a function of task descriptors, that is, query points, is needed. Using generalization, a task can be executed at an arbitrary query point $\boldsymbol{c}$ within the learned query space.

For statistical generalization, we use Gaussian process regression (GPR), which can be used to learn a function

$$F : c \mapsto [w_q, g_q, w_\tau, v] \tag{17}$$

A CMP, defined by w_q, g_q, w_τ, and v, can be used to execute a task, defined by a query c, in a compliant manner. By the above definition, F computes appropriate CMP parameters at the given query c, that is, at a given task variation.

Once the Gaussian process is *trained* using example training sets of CMPs, new appropriate CMPs for given queries c can be calculated by simple matrix multiplications, which can easily be accomplished in real time.

4. Evaluation

To evaluate the CMPs, a robotic arm with a mounted hand was used. The Kuka LWR-IV anthropomorphic arm was used. It has seven degrees of freedom and, important for the presented approach, torque sensors at each joint. We omit further details, for example, internal parameters, of the robotic arm and refer the reader to [8]. In order to grasp objects, a three fingered BarrettHand BH8-280 was mounted at the end of the robotic arm. Further details on grasping are omitted, as this is not the focus of the chapter. While the approach based on the CMPs does not need the dynamical model of the robot, the dynamical model of the Kuka robot was used in all experiments. Note that the controller for the Kuka robot does not allow to fully

disengage the dynamical model. However, using the CMPs, any possible model errors are compensated.

Evaluation of CMPs first focuses on estimating the quality of tracking the desired trajectory under various stiffness settings by measuring the errors between desired and actual robot trajectory. Here, the CMP approach was compared to a standard high-gain feedback control approach. Next the behavior of both systems when colliding with an unforeseen object was evaluated. The last part is concerned with evaluation of autonomously generated CMPs using the statistical graph search and evaluation of generalized CMPs.

4.1. Tracking accuracy evaluation

The analysis of tracking was performed on a pick-and-place task. The performance of the approach based on CMPs was compared with a common feedback approach. The evaluation setup can be seen in **Figure 6**, where a snapshot shows an example of pick-and-place movement. The initial movement trajectory for pick and place was demonstrated using kinesthetic guidance and encoded in DMPs as a part of CMPs. As the DMPs have been studied previously, we omit their specific evaluation and refer the reader to [13, 14, 24]. The demonstrated trajectory was then executed several times, using stiff robot control, that is, high feedback gains, which ensure accurate trajectory tracking. While executing the motion, the corresponding torques were obtained and encoded as CMPs. For evaluation, the mass of the object that the robot was holding was changed from 0.5 kg up to 4.5 kg with a step of 1 kg.

The movement was then executed using the complete CMPs, that is, including feed-forward torques, under different feedback gain settings. For comparison, for each object mass, several executions were performed by varying feedback gain parameters without using feed-forward torques, that is, using a common stiffens controller. To identify the tracking accuracy of the controllers, the maximum tracking error for each task execution was calculated. The maximal tracking error is defined as

$$e_m = \max_t \left(\| \boldsymbol{p}_a(t) - \boldsymbol{p}_d(t) \| \right), \tag{18}$$

Where $\mathbf{p_a}(t)$ is the current robot position on the trajectory, and the $\mathbf{p}_d(t)$ is the desired position of the trajectory, both in Cartesian space. The task was performed for several different feedback gains settings going from 50 Nm/rad up to 2000 Nm/rad, selected so that covered a wide spectrum of compliance exhibited by a Kuka LWR-IV robot.

Results of the evaluation are shown in **Figure 3**, where we can see the mean and standard deviation of the e_m over all feedback gain settings and for each object weights. We can clearly see here that the tracking accuracy is much larger if only feedback control is used compared to the novel approach based on CMPs. The plot also shows the point where the tracking error starts to increase notably, regardless of the approach, which was 50 Nm/rad. Based on this results, the feedback gains for future evaluation were set to K = 50 Nm/rad.

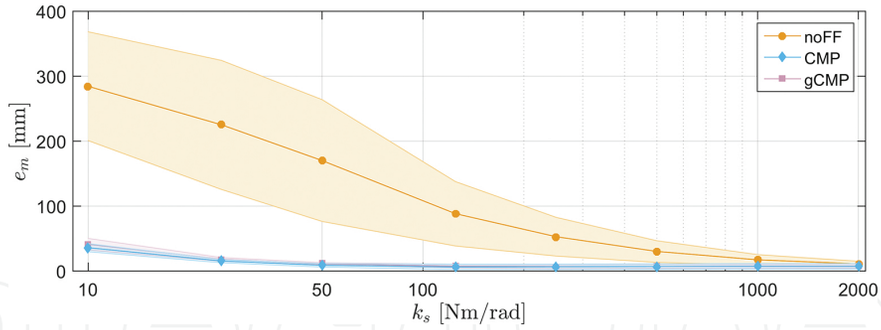

Figure 3. Mean and standard deviation of task's maximum error for different the object masses. The CMP indicates the performance CMPs system, while the noFF indicates the mean and standard deviation for the feedback control without feed-forward torque signal. The gCMP indicates the performance of the generalized CMPs trajectories.

4.2. Contact evaluation

To evaluate the behavior of CMP-based control approach, while colliding with an unforeseen object, or an environment, a downward motion was demonstrated first, and then executed to train the CMP. During learning the movement was executed in a free space, without any contacts. To evaluate the forces and behavior when robot collides with an unforeseen object or environment, an object was placed in the path of the robot. The behavior of the robot when colliding was analyzed for three different cases: impedance control with high gains (K = 2000 Nm/rad), impedance control with low gains (K = 50 Nm/rad), and previously learned CMP with (K = 50 Nm/rad).

The results of impact are shown in **Figure 4**, where we can see that in case of high feedback gains the tracking error remains relatively small thorough the movement. However, in this

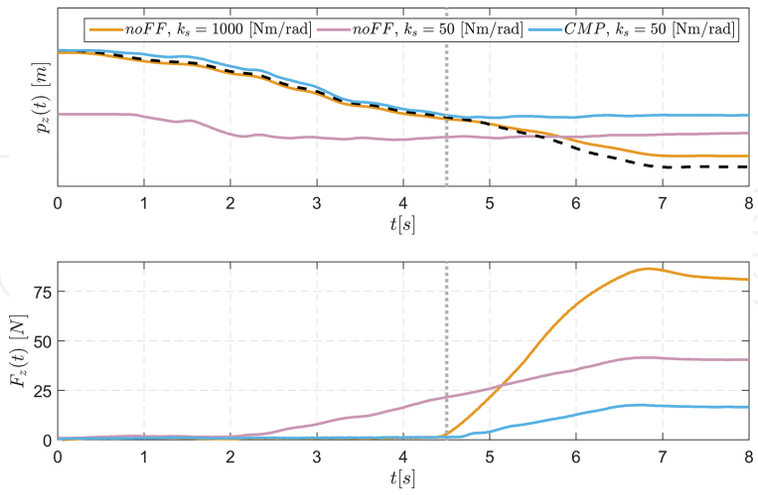

Figure 4. Robot colliding with an object while using different stiffness settings and control approaches. The graphs present collision trajectories and forces under two different stiffness settings (K = 1000 Nm/rad and K = 50 Nm/rad) and with two different control approaches (noFF and CMP). The top plot shows the actual robot task space position in vertical (z) axis, and the bottom plot shows TCP forces in the vertical (z) axis.

case, the forces rise significantly when the robot is in the contact with the environment. Vice versa, if the robot is compliant, that is, if the feedback gains are low, the forces are small, but the tracking accuracy is poor. Finally, CMPs combine both positive aspects: good tracking accuracy before contact and low interaction forces once the contact is established.

4.3. Hierarchical graph evaluation

Hierarchical graph approach was also evaluated using a Kuka LWR-IV robot arm. The demonstrator taught the robot several reaching movements while kinesthetically guiding the arm. Two examples of the learned movements that intersect each were shown. DMPs were used to encode all kinematic trajectories. They were used in the second step to obtain corresponding torque control signals, as described previously. Each movement was executed three times with different task time multipliers $\kappa = \{1,2,3\}$. The learned movement trajectories $\boldsymbol{q}_d$ and the corresponding torque control signals $\boldsymbol{\tau}_f$ were used to execute the learned reaching CMP using a low-gain feedback controller.

As described in Section 3.1, the database was built using learned CMPs and used to find new reaching movements. 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 was then enhanced with mean torques and corresponding time duration three times, once per task time multiplier. Using DMPs, a complete representation of new reaching movement's trajectories was synthesized. A close-up of transitions of two example demonstrated and newly synthesized movements is shown in **Figure 5**. Smooth and continuous transitions can be observed. In order to evaluate transitions of corresponding torque signals, tracking error was observed during executions of example demonstrated CMPs and newly generated CMPs. No significant rise in errors could be observed.

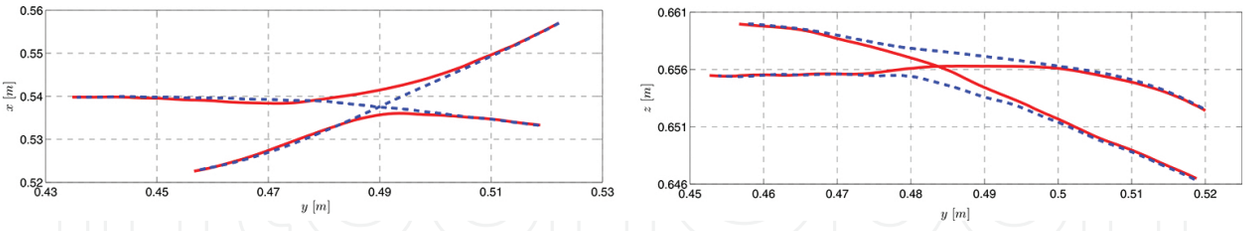

Figure 5. Smooth and continuous transition can be observed in a close-up of two example demonstrated and newly synthesized movements. Original demonstrated movements are denoted by red color, while newly synthesized are marked with a dashed blue line.

4.4. Statistical generalization evaluation

Evaluation of statistical generalization can be done by comparing generalized CMPs to learned non-generalized CMPs. The selected task was a pick-and place task. Evaluation setup can be seen in **Figure 6**. A trajectory $\mathbf{q}_S(\mathbf{t})$, which successfully moves a hand weight from the initial position to the final position, was demonstrated by kinesthetic guidance. It was then executed

five times using a standard high-gain controller which ensures high tracking accuracy. Mass of the object was changed each time. In this way, five different CMPs were learned, each having the same kinematic component and a different dynamic one. This set was used with statistical generalization techniques in order to generalize them over a one dimensional query, that is, a varying object mass. Generalized CMPs could compliantly move an object with arbitrary mass within the training space. New, generalized, CMPs were executed for nine different queries, covering demonstrated weights as well as points in between. Tracking errors were recorded for each execution. Each new generalized CMP was executed at eight different stiffness settings. Tracking errors were recorded for each execution. By comparing the maximum tracking errors of generalized CMPs to maximum tracking errors gained by executing learned CMPs, a slight but not statistically significant, increase in tracking error was observed. The errors prevailed at lower stiffness settings as the system is more susceptible to inaccuracies in generalized torque signals.

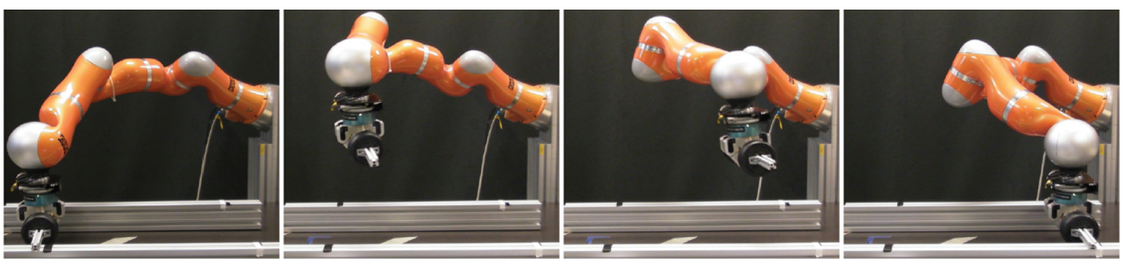

Figure 6. Experimental set-up for statistical generalization evaluation. The robot executed a pick-and-place task. Each execution the object weight was varied.

5. Discussion

In order to be efficient, robots need to autonomously react to the changes of the external conditions, such as changes in the environment or in the task. Relying on pure reproduction of learned motion will not provide the robots sufficient possibilities to reach to different situations, as learning to cover for all situations is simply impossible—the number of possibilities is far too great.

In this aspect, both presented methods of autonomous CMP generation, either through a hierarchical database search or through statistical generalization, expand the realm of learned motions to new, previously unexplored solutions. While the first method combines parts of demonstrated trajectories, the second one generates completely new trajectories within the database of learned motions. These new trajectories resemble the previously learned ones, that is, they have similar properties. The challenge of generating completely new motions using CMPs remains an open issue. While explorative methods, such as reinforcement learning methods, could be used to learn completely new motions with respect to a given reward, such methods are also known for requiring a large number of iterations. These might make practical use less convenient.

As an alternative to CMPs, dynamical models could be used. Here, it is important to note that CMPs model the complete task, for example, the task of interaction with a deformable object. Obtaining a dynamical model of a robot might be feasible, specifically for an expert, but obtaining a dynamical model for numerous tasks, such as the aforementioned task of interaction with a deformable object, might be extremely difficult, if not impossible. Thus, CMPs offer the means to solve this problem, as modeling is not needed, but they tend to solve it for one task at a time. Again, the presented methods of adapting to the changes of the task allow for learning of complete families of tasks from a small set of demonstrations.

Obtaining the CMPs is another area which requires further research, as currently these are obtained in a supervised manner during stiff robot behavior. Thus, the effort for the user still remains considerable. While we have proposed initial solutions in autonomously obtaining torque profiles in [25], a general method still remains an open research issue.

In conclusion, we have presented the CMP framework and two methods that expand the realm of possible application beyond simple imitation. The CMP framework, which bypasses the need for accurate dynamical modeling, was extensively evaluated and has been shown to offer a viable opportunity for compliant and at the same time accurate robotic behavior.

Acknowledgements

The research leading to these results has been partially founded from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under Grant agreement no. 600578, ACAT.

Author details

Miha Deniša*, Tadej Petrič, Andrej Gams and Aleš Ude

*Address all correspondence to: miha.denisa@ijs.si

Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenija

References

[1] Albu-Schaffer A., Eiberger O., Grebenstein M., Haddadin S., Ott C., Wimbock T., Wolf S., and Hirzinger G. Soft robotics. IEEE Robotics and Automation Magazine. 2008;15(3):20–30.

[2] Franklin D.W., and Wolpert D.M. Computational mechanisms of sensorimotor control. Neuron. 2011;72(3):425–442.

[3] Nguyen-Tuong D., and Peters J. Learning robot dynamics for computed torque control using local Gaussian processes regression. In: ECSIS symposium learning and adaptive behaviors for robotic systems; Edinburgh, Scotland, UK. 2008. p. 59–64.

[4] Nguyen-Tuong D., and Peters J. Model learning for robot control: a survey. Cognitive Processing. 2011;12(4):319–340.

[5] Bristow D.A., Tharayil M., and Alleyne A.G. A survey of iterative learning control. IEEE Control Systems. 2006;26(3):96–114.

[6] Schwarz M., and Behnke S. Compliant robot behavior using servo actuator models identified by iterative learning control. In: Proceedings of 17th robo cup international symposium; Eindhoven, Netherlands. 2013. p. 207–218.

[7] Gautier M., Jubien A., and Janot A. Iterative learning identification and computed torque control of robots. In: IEEE/RSJ international conference on intelligent robots and systems (IROS); Tokyo, Japan. 2013. p. 3419–3424.

[8] Bischoff R., Kurth J., Schreiber G., Koeppe R., Albu-Schaeffer A., Beyer A., Eiberger O., Haddadin S., Stemmer A., Grunwald G., and Hirzinger G. The KUKA-DLR Lightweight Robot arm—a new reference platform for robotics research and manufacturing. In: 41st international symposium on robotics; 2010.

[9] Petrič T., Gams A., Žlajpah L., and Ude A. Online learning of task-specific dynamics for periodic tasks. In: Intelligent robots and systems (IROS 2014); Chicago, IL, USA. 2014. p. 1790–1795.

[10] Deniša M., Gams A., Ude A., and Petrič T. Learning compliant movement primitives through demonstration and statistical generalization. IEEE/ASME Transactions on Mechatronics. 2015; (99):1.

[11] Deniša M., Petrič T., Asfour T., and Ude A. Synthesizing compliant reaching movements by searching a database of example trajectories. In: International conference on humanoid robots; Atlanta, GA, USA. 2013. p. 540–546.

[12] Krakauer J. W., Ghilardi M.F., and Ghez C. Independent learning of internal models for kinematic and dynamic control of reaching. Nature Neuroscience. 1999;2(11):1026–1031.

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

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

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

[16] Calinon S. A tutorial on task-parameterized movement learning and retrieval. Intelligent Service Robotics. 2015;9(1):1–29.

[17] Rose C., Cohen M.F., and Bodenheimer B. Verbs and adverbs: multidimensional motion interpolation. IEEE Computer Graphics and Applications. 1998;18(5):32–40.

[18] Kovar L., Gleicher M., and Pighin F. Motion Graphs. ACM Transactions on Graphics. 2002;21(3):473–482

[19] Yamane K., Revfi M., and Asfour T. Synthesizing object receiving motions of humanoid robots with human motion database. In: IEEE international conference on robotics and automation (ICRA); Karlsruhe, Germany. 2013. p. 1621–1628.

[20] Yamane K., Yamaguchi Y., and Nakamura Y. Human motion database with a binary tree and node transition graphs. Autonomous Robots. 2010;30(1):87–98.

[21] Deniša M., and Ude A. Synthesis of new dynamic movement primitives through search in a hierarchical database of example movements. Int J Adv Robot Syst. 2015;12(137): 1–13

[22] Calandra R., Ivaldi S., Deisenroth M., and Peters J. Learning torque control in presence of contacts using tactile sensing from robot skin. In: IEEE-RAS 15th international conference on humanoid robots (Humanoids); Madrid, Spain. 2015. p. 690–695.

[23] Steinmetz F., Montebelli A., and Kyrki V. Simultaneous kinesthetic teaching of positional and force requirements for sequential in-contact tasks. In: IEEE-RAS 15th international conference on humanoid robots (Humanoids); Madrid, Spain. 2015. p. 202–209.

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

[25] Petrič T., Colasanto L., Gams A., Ude A., and Ijspeert A.J. Bio-inspired learning and database expansion of compliant movement primitives. In: Humanoid robots (Humanoids); Seoul, Korea. p. 356–351.

[26] Gams A., Ijspeert A.J., Schaal S., and Lenarčič J. On-line learning and modulation of periodic movements with nonlinear dynamical systems. Autonomous Robots. 2009;27(1):3–23.

IntechOpen