Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives

Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto

***Abstract*—Acquisition of new sensorimotor knowledge by imitation is a promising paradigm for robot learning. To be effective, action learning should not be limited to direct replication of movements obtained during training but must also enable the generation of actions in situations a robot has never encountered before. This paper describes a methodology that enables the generalization of the available sensorimotor knowledge. New actions are synthesized by the application of statistical methods, where the goal and other characteristics of an action are utilized as queries to create a suitable control policy, taking into account the current state of the world. Nonlinear dynamic systems are employed as a motor representation. The proposed approach enables the generation of a wide range of policies without requiring an expert to modify the underlying representations to account for different task-specific features and perceptual feedback. The paper also demonstrates that the proposed methodology can be integrated with an active vision system of a humanoid robot. 3-D vision data are used to provide query points for statistical generalization. While 3-D vision on humanoid robots with complex oculomotor systems is often difficult due to the modeling uncertainties, we show that these uncertainties can be accounted for by the proposed approach.**

***Index Terms*—Active vision on humanoid robots, humanoid robots, imitation learning, learning and adaptive systems.**

I. Introduction

LEARNING of behaviors that can be applied to solve a given task, regardless of the current configuration of the external world, is a difficult problem because the search space that needs to be explored is potentially huge [1]. The size of the search space depends both on the number of degrees of freedom of the robot and on the objects involved in the action. Furthermore, external objects can affect the search space indirectly. To overcome problems arising from high-dimensional and continuous perception-action spaces, it is necessary to guide the search process. One of the most successful paradigms that can be used for this purpose is imitation or robot programming by demonstration [1], [2].

Robot programming by demonstration requires the acquisition of example trajectories, which can be captured in various ways. Motion capture techniques based on optical- or magnetic-tracking devices were applied successfully to replicate complex movements on humanoid robots such as dancing, which would be difficult to program manually [3]–[6]. Computer vision techniques that aim to capture human motion without special markers are slowly getting matured [7], but the application of specialized tracking devices is still preferred in most of the recent works in order to avoid the pitfalls of computer vision approaches. Alternatively, a robot can be physically guided through the desired trajectory, which is recorded proprioceptively. This method requires that the robot is back-drivable [8], [9] or can compensate for the influences of external forces [10]–[12]. The advantage of kinesthetic guiding is that the movements are recorded directly on the learning robot and do not need to be first transferred from a system with different kinematics and dynamics.

A simple reproduction of the trained movements is not useful for problems that involve the manipulation of objects because in such tasks the observed movements need to be adapted to the current state of the 3-D world. Already, early research on learning from demonstration has stressed the importance of task segmentation and the extraction of meaningful action units [13]–[15].

The main interest of this paper is in the generalization of the available action knowledge regardless of how this knowledge was initially acquired. A methodology that can adapt single trajectories obtained by imitation was proposed by Miyamoto *et al.* [16]. They made use of a spline-based representation for the desired trajectory, which they referred to as via points. By monitoring its performance, the robot was able to continuously adapt the via points until it could play a fairly difficult Japanese game of kendama or execute tennis serves. This type of approach has later been studied in the frame of modern reinforcement learning theories [9]. One of the main issues is the speed of convergence, which is important for online applications of reinforcement learning and is an active topic of research [17], [18].

Manuscript received September 4, 2009; revised February 22, 2010 and July 1, 2010; accepted August 4, 2010. Date of current version September 27, 2010. This paper was recommended for publication by Associate Editor A. Ijspeert and Editor J.-P. Laumond upon evaluation of the reviewers' comments. This work was supported in part by the European Union Cognitive Systems Project PACO-PLUS under Grant FP6-2004-IST-4-027657. The first author (A. Ude) was also supported by the National Institute of Information and Communications Technology within the JAPAN TRUST International Research Cooperation Program.

A. Ude is with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, 1000 Ljubljana, Slovenia, and also with the Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International, Kyoto 619-0288, Japan (e-mail: ales.ude@ijs.si).

A. Gams was a Visiting Researcher at Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International, Kyoto 619-0288, Japan, in summer 2009. He is now with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, 1000 Ljubljana, Slovenia (e-mail: andrej.gams@ijs.si).

T. Asfour is with the Institute for Anthropomatics, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: asfour@kit.edu).

J. Morimoto is with the Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International, Kyoto 619-0288, Japan (e-mail: xmorimo@atr.jp).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes seven QuickTime format movie clips, which show the results on real robots. The complete material is 45.2 MB in size. Contact ales.ude@ijs.si for further questions about this work.

Digital Object Identifier 10.1109/TRO.2010.2065430

Numerous representations have been proposed to encode robot trajectories for learning. Spline-based representations like the one utilized in [16] have been employed in robotics in many contexts [19], but their explicit dependency on time can be cumbersome [20]. Hidden Markov models (HMMs) are another popular methodology to encode and generalize the observed trajectories [21]–[23]. It has been shown that HMMs can be used effectively for motion and action recognition [23] and to determine which control variables should be imitated and how [22]. Kulić *et al.* [24], [25] extended these works by showing how to use HMMs to automatically cluster large databases of movements into the constituent motor primitives. Gaussian mixture models [26] provide another motor representation with some advantages over HMMs during reproduction of the learned movements. Calinon *et al.* [26] also demonstrated how movements can be generalized by taking into account different analytically specified constraints between hands and objects. Yamane *et al.* [27] proposed a graph-theoretical approach that organizes the captured data in a large motion database. In their work, the problem of motion generation becomes a search problem.

A fundamentally different approach to motion representation based on nonlinear dynamic systems as policy primitives was proposed in [28] and [29]. The resulting control policies were termed as *dynamic movement primitives* (DMPs). DMPs are based on systems of second-order differential equations, which encode the properties of the desired motion. Ijspeert *et al.* [28], [29] developed equations for periodic and discrete movements and demonstrated the learning of tasks such as tennis strokes and drumming. One of the most important advantages of DMPs is the ability to take into account perturbations and to include feedback terms. Feedback terms can be added to change the timing [20] and/or avoid some areas of the workspace. Another approach based on the idea of dynamic systems of Ijspeert *et al.* is described in [8], where the final trajectory is generated as a linear combination of attractor dynamics and Gaussian mixture models that were learned from example trajectories.

A. Contribution of This Paper and Related Work

The main purpose of this paper is to propose and experimentally evaluate a methodology for generalization of example trajectories to new situations that were not observed during training, using a representation suitable for robot control. To this end, every example trajectory is associated with parameters that describe the characteristics of the task, typically its goal, and serve as query points into an example database. For instance, in our experiments, we studied the problem of ball throwing toward a specified target. In these experiments, we first acquired a set of example robot trajectories that resulted in ball throws toward different targets. The task of generalization is to synthesize throws toward any given target within the training space. The parameters specifying the goal of the task are, in this case, the target positions.

Our work was inspired by motor-tape theories, in which example movement trajectories are stored directly in memory [30], [31]. Initial implementations of this theory were based on lookup tables [32] that cannot generalize. More modern versions interpret the task of generalization as a problem of fitting a multivariate function to the stored training data, where the fitted function must generalize to new situations [30]. Regression methods based on local weighting of training data at execution time can be applied for this purpose. They have proven to be sufficiently efficient for online control [33].

We utilize nonlinear dynamic systems as a basic motor representation. Other researchers have shown that by changing the underlying differential equations, DMPs can be modified in several ways to account for various perturbations that might occur during the execution of the task. For example, it has been suggested to add terms that enable obstacle avoidance to the basic DMP equations [34], [35]. Such modifications are very useful, but they are necessarily designed for one particular issue only. For each new problem, an expert must redesign the underlying dynamic system, which is unsatisfactory for robots that need to solve new problems every day. The approach proposed in this paper enables the generalization of DMPs to new situations using the available training movements and the goal of the task, which can normally be specified in a natural way. Since the generalized trajectories are encoded as standard DMPs, we are still able to apply analytical modifications of the underlying dynamic system at execution time in order to account for unforeseen perturbations, which are not part of the training data. In addition, Section III-C demonstrates that the proposed approach is suitable for online integration with a humanoid robot's active vision.

It is also possible to generalize the example movements to new situations by specifying constraints and weighting them with respect to their importance for the task [22]. For example, if the initial positions on the training trajectories are found to be less important, the robot can modify its motion to reach the final position from a different initial position at the expense of fidelity of reproduction. Systems like this require an analytical model to describe the constraints. The approach proposed in this paper requires no physical models of the task to be specified; generalization is done in a statistical way.

Another methodology for taking into account external variables was proposed in [36], where—instead of synthesizing the optimal DMP parameters at execution time as in our approach—the basic DMP equations were modified by coupling them with the perceptual input. This coupling provides additional parameters that can be used to learn a more globally valid representation as opposed to the local one generated by our system. As shown in [36], the external variables have a similar role as query points into the example database in our approach. It is, however, more difficult to ensure that such a system learns all interdependencies between the perceptual input and the DMP parameters because, in general, it is unknown how many additional degrees of freedom are needed. This problem is avoided by using local models. Our work is also related to the theories in which generalization takes place through a combination of motor primitives [37]. An architecture that linearly combines internal models, which can be regarded conceptually as motor primitives, based on the output of a responsibility estimator has been proposed in [38] and tested in simulation [39]. Work under progress has been reported in [18], where motor primitives are linearly combined based on the output of a gating network. Such an approach takes

generalization to a higher abstraction level away from the training data, which reduces the dimensionality of the problem but makes accurate generalization more difficult.

II. Action Generalization Using Dynamic Movement Primitives and Local Weighting

Let us assume that we have a set of example trajectories together with the parameters characterizing the task

$$\mathcal{Z} = \{y_{\mathrm{d}}^k(t_{k,j}), \dot{y}_{\mathrm{d}}^k(t_{k,j}), \ddot{y}_{\mathrm{d}}^k(t_{k,j}); \mathbf{q}_k | k = 1, \ldots, M \quad j = 1, \ldots, T_k\} \quad (1)$$

where $y_d^k(t_j),\ \dot{y}_d^k(t_j),$ and $\ddot{y}_d^k(t_j)$ are the measured positions, velocities, and accelerations on trajectory k, M is the number of examples, and T_k the number of sampling points on each trajectory. $\mathbf{q}_k \in \mathbb{R}^n$ are the parameters describing the task in a given example situation and are usually related to the goal of an action. In our system, they are used as query points into a database of example trajectories. Such data can be obtained either by kinesthetic guiding or from human demonstrations. The trajectories can be specified either in joint or task space. In a DMP formulation (see Appendix A for a brief review of the DMP representation), every degree of freedom is described by its own dynamic system but with a common phase to synchronize them. Indexing of the degrees of freedom is omitted from (1) for clarity. The issue is how to generate a DMP specifying a movement for every new query point $\mathbf{q}$, which, in general, will not be one of the example queries $\mathbf{q}_k$. As explained in Appendix A, DMPs are specified by parameters $\mathbf{w},\ \tau$ (or $\Omega = 1/\tau$ in case of periodic movements), and g. Thus, we need to learn a function

$$\mathbf{G}(\mathcal{Z}) : \mathbf{q} \longmapsto [\mathbf{w}^T, \tau,\ g]^T. \quad (2)$$

Examples $\mathcal{Z}$ specified in (1) are the data used for learning. While the problem of learning of g and τ has also been studied in [18], here, the complete learning problem is treated.

In general, the functional relationship between $\mathbf{q}$ and $[\mathbf{w}^T, \tau,\ g]^T$ given a set of examples $\mathcal{Z}$ is unknown. In most cases, it is difficult to find a global model that provides good approximation for the function $\mathbf{G}(\mathcal{Z})$. We therefore avoid global model identification and rather apply regression techniques to generalize the movements. Due to significantly different sizes of datasets involved in the calculation of parameters $\mathbf{w}$ on the one hand, and g and τ on the other hand, we propose to employ different methods to estimate them. More specifically, we applied locally weighted regression (LWR), which is a regression method that fits local models to nearby data [40], for the estimation of $\mathbf{w}$. LWR has a lower computational complexity than many other nonparametric regression methods including Gaussian process regression (GPR) [41]. On the other hand, due to its high accuracy, we utilized GPR [42] to estimate g and τ. Another advantage of GPR is that there is no need to determine the place and number of basis functions. However, its computational cost can be prohibitive especially in the case of online learning, where the dataset gradually increases in size. This was not a problem when learning g and τ in our experiments.

Our case studies include tasks such as reaching, ball throwing, and drumming. In the case of reaching movements, the goal of an action (or query point) was simply the final reaching destination in Cartesian space, whereas the trajectories and, consequently, the attractor points of the DMPs were defined in joint space. Thus, the query and attractor points were connected through the robot kinematics. In other cases, the queries were less directly associated with the parameters of the DMP. For example, in the case of ball throwing, the goal was characterized by the position of the basket into which the ball should be thrown. This position is not directly encoded in the parameters of the discrete DMP, as given by (25) and (26), shown later. As example of periodic movements, we studied drumming on a cymbal and a drum, where the height at which the cymbal was mounted varied. The robot adapted its drumming movements to the varying height of the cymbal. In this case, the height of the cymbal was used as query point, which is again not directly encoded in the periodic DMP equations (28) and (29), shown later. In summary, the query points normally originate from an intuitive characterization of the task, but the functional relationship between them and the DMP parameters may not be straightforward.

Note that $\mathbf{G}(\mathcal{Z})$ becomes a function only by constraining the solution trajectory to be as similar as possible to the example trajectories. For example, there are many different ways of how to throw a ball into a basket at a certain location. The relationship between the basket positions (query points) and DMP parameters as given in (2) becomes a function by requesting that the generalized throwing movements are similar to the example throws. The similarity criterion is embedded into the regression process in our practical implementation.

In the following, we provide a methodology to generate new DMPs for situations that are not part of the example database. To fully specify a DMP that defines a movement in a new situation, we need to estimate the shape parameters $\mathbf{w}$, goal g, and, respectively, the time constant τ in case of discrete movements and frequency Ω in case of periodic movements. Section II-A first explains the generation of DMPs using only one training trajectory. We continue the description of the estimation of parameters $\mathbf{w}$ for discrete (see Section II-B) and periodic movements (see Section II-C). Section II-D deals with the estimation of other DMP parameters that vary from example to example (g and τ or Ω). The rest of the DMP parameters ($\alpha_z,\ \beta_z$, and α_x) are fixed and are determined so that the convergence of the underlying dynamic system is guaranteed.

A. Reproduction From Single Demonstration

With the DMP representation of Appendix A, the trajectory of any smooth movement can be approximated by adapting the parameters w_i of (23) and (27). The system of two first-order linear equations (25) and (26) in the case of discrete movements and (28) and (29) in the case of periodic movements can be rewritten into one second-order equation by replacing z with $\tau\dot{y}$ in (25) and (28), respectively

$$\tau^2\ddot{y} + \alpha_z\tau\dot{y} - \alpha_z\beta_z(g - y) = f \quad (3)$$

with f defined as in (23) and (27). All of the above-mentioned equations are shown later in Appendix A. The formula $\tau = 1/\Omega$ is used in the case of periodic movements. Note that time constant τ must be the same for all degrees of freedom. In our experiments with discrete movements, we used $\tau = t_T$, where t_T is the duration of the training movement. On the other hand, the attractor points g vary across the degrees of freedom. They can be extracted directly from the data as $g = y_{\mathrm{d}}(t_T)$ in the case of discrete movements and $g = \sum_j y_{\mathrm{d}}(t_j)/T$ in the case of periodic movements. Writing

$$F(t_j) = \tau^2 \ddot{y}_{\mathrm{d}}(t_j) + \alpha_z \tau \dot{y}_{\mathrm{d}}(t_j) - \alpha_z \beta_z (g - y_{\mathrm{d}}(t_j))$$
$$\mathbf{f} = \begin{bmatrix} F(t_1) \\ \cdots \\ F(t_T) \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_1 \\ \cdots \\ w_N \end{bmatrix} \tag{4}$$

we obtain the following system of linear equations:

$$\mathbf{X}\mathbf{w} = \mathbf{f} \tag{5}$$

which needs to be solved to estimate the remaining parameters of a DMP that describe the desired motion. In the case of discrete movements, we have

$$\mathbf{X} = \begin{bmatrix} \frac{\Psi_1(x_1)}{\sum_{i=1}^N \Psi_i(x_1)} x_1 & \cdots & \frac{\Psi_N(x_1)}{\sum_{i=1}^N \Psi_i(x_1)} x_1 \\ \cdots & \cdots & \cdots \\ \frac{\Psi_1(x_T)}{\sum_{i=1}^N \Psi_i(x_T)} x_T & \cdots & \frac{\Psi_N(x_T)}{\sum_{i=1}^N \Psi_i(x_T)} x_T \end{bmatrix} \tag{6}$$

and in the case of periodic movements

$$\mathbf{X} = r \begin{bmatrix} \frac{\Gamma_1(\phi_1)}{\sum_{i=1}^N \Gamma_i(\phi_1)} & \cdots & \frac{\Gamma_N(\phi_1)}{\sum_{i=1}^N \Gamma_i(\phi_1)} \\ \cdots & \cdots & \cdots \\ \frac{\Gamma_1(\phi_T)}{\sum_{i=1}^N \Psi_i(\phi_T)} & \cdots & \frac{\Gamma_N(\phi_T)}{\sum_{i=1}^N \Gamma_i(\phi_T)} \end{bmatrix}. \tag{7}$$

Here, x_i and ϕ_i are obtained by respective integration of (24) and (30), shown later. The parameters $\mathbf{w}$ can be calculated from the solution of the above systems of linear equations in a least-squares sense.

While it is common to estimate discrete movements using single demonstrations and the aforesaid batch approach, periodic movements are more often estimated recursively by repeating the movement a number of times. Note that the integration of (28)–(30), shown later, requires that the frequency of movement Ω is known. Unlike time duration, this frequency is not directly observable and must be estimated. It has been suggested to replace (30), shown later, with the system of adaptive frequency oscillators [43], [44] (see also Appendix B) to automatically determine the frequency during demonstration. The waveform estimation can then be realized by solving the equation system (5) using recursive least squares with a forgetting factor

$$\mathbf{P}_j = \frac{1}{\lambda}\left(\mathbf{P}_{j-1} - \frac{\mathbf{P}_{j-1}\mathbf{x}_j\mathbf{x}_j^T\mathbf{P}_{j-1}}{\lambda + \mathbf{x}_j^T\mathbf{P}_{j-1}\mathbf{x}_j}\right) \tag{8}$$

$$\mathbf{w}_j = \mathbf{w}_{j-1} + (f_j - \mathbf{x}_j^T\mathbf{w}_{j-1})\mathbf{P}_j\mathbf{x}_j \tag{9}$$

where $\mathbf{P}_0 = \mathbf{I}$, $\mathbf{w}_0 = 0$, $f_j = F(t_j)$, $\mathbf{x}_j$ is the M-dimensional column vector associated with the corresponding row of the system matrix $\mathbf{X}$, $0 < \lambda \leq 1$ is the forgetting factor, and the final weights are given as $\mathbf{w} = \mathbf{w}_T$.

It is important to note that with this system, the frequency estimation should be done simultaneously with the recursive estimation of the form parameters $\mathbf{w}$; at each time t_j, first the current Ω is estimated by integrating (31)–(33), shown later. The estimated Ω is then used to calculate the target values $F(t_j)$ of (4), where again, $\tau = 1/\Omega$. Finally, the new estimate $\mathbf{w}_j$ is calculated using the recursion (8) and (9). Thus, training can be organized as a coaching process; the coach demonstrates the movement, the robot simultaneously executes the movement using the currently estimated parameters, and the coach stops the demonstration once he/she is satisfied with the robot's performance. This approach is similar to the idea proposed in [45], where the transfer of human motor skills to the robot is supported by a training system that keeps the human instructor and the robot in a real-time control loop. Note that (8) includes the forgetting factor λ; thus, later training data have more influence on the estimated parameters than earlier data.

In the aforesaid equations, α_x, α_z, and β_z are constant. They are set so that the convergence of the underlying dynamic system is ensured [20]. Unlike in some other works where the parameters w_i are estimated independently of each other [28], [29], we apply a full linear system (5) to estimate $\mathbf{w}$. In this way, we can approximate trajectories more accurately because we can take into account the interplay between the neighboring basis functions Ψ_i of (23) and (27), shown later, which benefits generalization. Note that the separate estimation of $\{w_i\}$ has its advantages, especially in the presence of noise when overfitting can become a problem or when $\{w_i\}$ are used for classification [29]. However, here the example trajectories are trained movements suitable for execution on a robot. They are therefore relatively noise-free, which alleviates the danger of overfitting. It was therefore sufficient in our experiments to determine the parameters c_i and h_i by setting the distribution pattern[1] and increasing N until the desired reconstruction accuracy on all example trajectories was achieved. More advanced methods [46] could be applied to determine N, c_i, and h_i if overfitting became a problem.

B. Generalization of Discrete Movements

In one-shot learning of Section II-A, all data are relevant for the estimation of shape parameters $\mathbf{w}$. Such global optimization approaches are problematic for generalization from multiple examples because, in general, no global task models are available. Control policies that solve the task in situations, which are very different from the current one, do not carry much information about the optimal policy in the current situation. Taking the example of ball throwing, it is reasonable to assume that the trajectories associated with target positions close to the current

[1]For a given N, $c_i = \exp(-\alpha_x \frac{i-1}{N-1})$, $h_i = \frac{2}{(c_{i+1}-c_i)^2}$, $h_N = h_{N-1}$, $i = 1, \ldots, N$.

target are more relevant than example trajectories associated with more distant targets.

The synthesis of shape parameters $\mathbf{w}$ is based on the sampled trajectory points included in the training dataset (1). Since the number of sampling points is usually very large, it is beneficial to apply a method with relatively low computational complexity, such as, for example, locally weighted regression. In an LWR setting and for a given query point $\mathbf{q}$, the optimal parameters can be calculated directly from the available data by minimizing the objective function

$$C(\mathbf{w};\mathbf{q}) = \sum_k L(\Xi(\mathbf{q}_k,\mathbf{w}),\mathbf{f}_k) K(d(\mathbf{q},\mathbf{q}_k)). \quad (10)$$

Based on (5), local models are characterized by

$$L(\Xi(\mathbf{q}_k,\mathbf{w}),\mathbf{f}_k) = \|\mathbf{X}_k\mathbf{w}-\mathbf{f}_k\|^2 \quad (11)$$

where $\Xi(\mathbf{q}_k,\mathbf{w}) = \mathbf{X}_k\mathbf{w}$. $\mathbf{X}_k$ needs to be calculated as in (6) for discrete or in (7) for periodic movements. $\mathbf{f}_k$ can be computed as in (4) but using the generalized τ and g (see Section II-D). We need to minimize the objective function

$$\sum_{k=1}^{M} \|\mathbf{X}_k\mathbf{w}-\mathbf{f}_k\|^2 K(d(\mathbf{q},\mathbf{q}_k)) \quad (12)$$

with respect to $\mathbf{w}$. Here, K is the kernel function and d is the metrics in the space of query points $\mathbf{q}$.

There are many possibilities to select the weighting kernel K [40]. We chose the tricube kernel

$$K(d) = \begin{cases} (1-|d|^3)^3, & \text{if } |d|<1 \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

because this kernel has finite extent and a continuous first and second derivative, which means that the first two derivatives of the prediction (as a function of query points) are also continuous. The finite support of K reduces the computational complexity of the optimization problem (12) because the example trajectories for which K vanishes do not influence generalization. In this way, we reduce the size of the system matrix associated with the objective function (12). As discussed in [40] and [47], the choice of weighting function is rarely critical for the performance of locally weighted regression. The selected kernel performed well in our experiments.

K and distance d in the space of query points determine how much influence each of the example movements has on the final estimate of the control policy. The influence of each example movement should diminish with the distance of the query point $\mathbf{q}$ from the data point $\mathbf{q}_k$. A standard weighted Euclidean distance can be used when query points are given in Euclidean space

$$d(\mathbf{q},\mathbf{q}_k) = \|\mathbf{D}(\mathbf{q}-\mathbf{q}_k)\|,\ \mathbf{D} = \mathrm{diag}(1/a_i),\ a_i > 0. \quad (14)$$

Ideally, $\mathbf{D}$ should be determined so that the error in the execution of the task is as small as possible. To reduce the resulting problem to a 1-D optimization problem while still accounting for the possibly varying spacing across dimensions of the query point space, we defined

$$a_i = c * \max_{j=1,\ldots,M} \min_{k=1,\ldots,M} \{|q_{j,i}-q_{k,i}\}. \quad (15)$$

In this way, $\mathbf{D}$ is fully specified by a parameter $c \in \mathbb{R}$, which we calculated by minimizing the validation set error [40] (see also Section III-D). Such an approach is sufficient if the data are approximately equally distributed along the coordinates axes, as it was in all our experiments. The validation set-error method evaluates the difference between the predicted output and the observed value in the validation set, where the data in the validation set are not used for training. The minimization of the validation-set error gave better results on regularly spaced training data than leave-one-out cross-validation method because it could better take into account the in-between query points. In all of our experiments, the data were approximately regularly spaced and we obtained the optimal value $c \approx 2.2$. With such c, the number of example trajectories with nonzero $K(d(\mathbf{q},\mathbf{q}_k))$ was about 4^n.

The computational complexity of solving the least-squares system (12) is $\mathcal{O}(N^2T)$, $T \leq \sum_{k=1}^{M} T_k$ and thus increases linearly with the number of data points considered by LWR and quadratically with the number of radial basis functions used in (23) and (27), shown later, respectively. Due to our choice of weighting kernel K, we normally have $K(d(\mathbf{q},\mathbf{q}_k)) = 0$ for many k. Moreover, by cutting the support of basis functions (23) and (27), shown later, once their value falls below a certain threshold, matrices $\mathbf{X}_k$ become sparse as well. The quadratic dependence on the number of basis functions is not a problem because this number is generally much lower than the number of data points. In most of our experiments, there were around 10 000–50 000 data points and 25–50 basis functions for DMPs. These facts make computational complexity sufficiently low to resolve the least-squares problem (12) online using standard methods from sparse-matrix algebra.

C. Generalization of Periodic Movements

In Section II-A, we describe a method that estimates the parameters of a DMP to reproduce a demonstrated periodic trajectory. The simultaneous and recursive estimation of frequency Ω and form parameters $\mathbf{w}$ allows the robot to reproduce the demonstrated motion immediately, which is important for coaching. To generalize the learned movements to new situations, we store the estimated frequencies and the sampled movements from the last few movement periods, which produced a good movement on the robot as judged by the teacher. In our experiments, we used the last five periods.

The aforementioned training process makes the following data available for generalization purposes: Trajectory data points within the last few periods of motion $\{y_{\mathrm{d}}^k(t_{k,j}), \dot{y}_{\mathrm{d}}^k(t_{k,j}), \ddot{y}_{\mathrm{d}}^k(t_{k,j}) | k=1,\ldots,M, \quad j=1,\ldots,T_k\}$ and the associated frequencies Ω_k. Each trajectory is also associated with query parameters $\mathbf{q}_k \in \mathbb{R}^n$. Given a new query point $\mathbf{q}$, we could now directly the minimize objective function (12) to calculate the new parameters $\mathbf{w}$. However, since the data were acquired by recursive least squares with forgetting factor λ, it is necessary to generalize recursively as well. In this way, we ensure that the generalization process discounts earlier training data in the same way as the coaching process. To achieve this, we need to parse all of the sampled

procedure GeneralizePeriodicMotion
  $\mathbf{P} = \mathbf{I},\ \mathbf{w} = 0;$
  $\forall k,\ \phi_k = t_1\Omega_k,\ j_k = 1;$
  while not all data points processed
    $k' = \arg\min_k\{\phi_k(t_{k,j_k})\};$
    $\phi_{k'} = t_{k',j_{k'}}\Omega_{k'};$
    using $y_{\mathrm{d}}^{k'}(t_{k',j_{k'}}), \dot{y}_{\mathrm{d}}^{k'}(t_{k',j_{k'}}), \ddot{y}_{\mathrm{d}}^{k'}(t_{k',j_{k'}}),\ \phi_{k'}$
    calculate
      $\mathbf{x} = \mathbf{x}_{k'}K(d(\mathbf{q}, \mathbf{q}_{k'}));\ f = F(t_{k',j_{k'}})K(d(\mathbf{q}, \mathbf{q}_{k'}));$
    update $\mathbf{P}$ and $\mathbf{w}$ using Eqs. (8)–(9) with the calculated $\mathbf{x}$ and f;
    $j_{k'} = j_{k'} + 1;$
  end

Fig. 1. Procedure for the generalization of periodic movements.

trajectory points in a proper order. For this purpose, it is necessary to maintain a separate phase information for every example trajectory to account for different frequencies during the generalization process. The resulting procedure shown in Fig. 1 ensures that the data from all of the trajectories are parsed and discounted across example trajectories in the same way as during coaching.

D. Estimation of Attractor Points, Timings, and Frequencies

Unlike α_x, α_z, β_z, and N, which are kept constant across the example trajectories during generalization, time constant τ in the case of discrete movements and frequency Ω in the case of periodic patterns as well as the attractor points $g_i, i = 1, \ldots, D$, where D is the dimension of the space in which the motion is specified, change from example to example. We extract these parameters from the training data (see Section II-A), which enables us to estimate the function that transforms query points $\mathbf{q}$ into τ, Ω, and $g_i, i = 1, \ldots, D$, directly.

Among various paradigms that could be used for this purpose, Gaussian process regression (GPR) has proven to be especially effective. It is a Bayesian regression method that provides a predictive distribution. GPR exhibits good generalization performance and the predictive distribution can be used to measure the uncertainty of the estimated function. Complexity of the model can be adaptively changed according to the newly acquired sampled data. There is no need to explicitly place additional basis functions and parameters. Simultaneously, GPR can avoid the danger of overfitting. It has been shown that this technique outperforms other regression methods on some relevant robotic problems such as estimating inverse dynamics of a seven degrees of freedom robotic arm [42]. Since the number of training data points is much lower when estimating τ, Ω, and g_i than when estimating the form parameters $\mathbf{w}$ (now it is the same as the number of training trajectories, before we had to consider all of the sampled points on the example trajectories), the computational complexity is not a problem and we selected GPR for the estimation of these parameters. A brief review of GPR is given in Appendix C.

With GPR, new estimates are calculated using (36) from the Appendix. The most computationally expensive part of this formula is the calculation of $[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}]^{-1}$, but since this matrix depends only on the training data, the necessary calculations can be done off-line using, for example, the Cholesky decomposition. Note that by writing

$$\mathbf{z} = [\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y} \tag{16}$$

(36) and the new parameters $\bar{y}^*$, i.e., $\bar{\tau}^*$, $\bar{\Omega}^*$, and $\bar{g}_i^*$, associated with the query $\mathbf{Q}^* = \mathbf{q}^*$ can be written as

$$\bar{y}^* = \sum_{k=1}^{M} k(\mathbf{q}^*, \mathbf{q}_k)z_k. \tag{17}$$

Thus, similarly as in LWR technique of (12), the data are weighted based on the distance between training query points and the current query point; hence, nearby training points influence the result more. To generate a new movement, the robot is given a desired query point. Using GPR, g_i and τ (or Ω) for this query are calculated. The optimal parameters $\mathbf{w}$ are then estimated using the algorithms of Section II-B or C. The sketch of the complete training and generalization procedure is given in Fig. 2.

procedure CollectTrainingData
  acquire trajectory points $\{y_{\mathrm{d}}^k(t_{k,j}), \dot{y}_{\mathrm{d}}^k(t_{k,j}), \ddot{y}_{\mathrm{d}}^k(t_{k,j})|$ $k = 1, \ldots, M,\ j = 1, \ldots, T_k\}$, e. g. by kinesthetic guiding or direct imitation;
  extract the attractor points $\mathbf{g} = \{g_k\}$ and time constants $\tau = \{\tau_k\}$ or frequencies $\mathbf{\Omega} = \{\Omega_k\}$, $k = 1, \ldots, M$;
  associate the acquired trajectories with query points $\mathbf{Q} = \{\mathbf{q}_k\}$;
  calculate the Cholesky decomposition of covariance matrices $\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}$;

procedure GeneralizeTrajectory
  for a given query point $\mathbf{Q}^* = \mathbf{q}^*$ and using Gaussian process regression (36), **estimate** the attractor point g^* and time constant τ^* or frequency Ω^* (with $\mathbf{y} = \mathbf{g}$, τ, $\mathbf{\Omega}$, respectively);
  minimize (12) to estimate the parameters $\mathbf{w} \in \mathbb{R}^N$ specifying the generalized DMP for query point $\mathbf{q}^*$;

Fig. 2. Training and generalization of goal-directed actions (for one dimension of the space in which trajectories are defined).

III. Experiments

We conducted a number of experiments to demonstrate the usefulness of the proposed approach. The following tests were done for discrete movements: 1) a reaching study in simulation and on a real robot to evaluate the accuracy of reproduction when learning discrete movements, 2) experiments with a full-size humanoid robot that connect the proposed approach with active vision and also demonstrate grasping, and 3) a simulated and a real-world ball-throwing study that confirm that also more dynamic aspects of the task can be generalized. To validate the generalization of periodic movements, we conducted 1) a simulated periodic-pattern-movement study and 2) a real-world drumming experiment.

Two humanoid robots were used to validate the proposed approach: a small humanoid robot HOAP-3 built by Fujitsu Automation and a full-size humanoid robot CB-i built in collaboration between Sarcos, ATR, and JST [48]. Both robots enable the acquisition of trajectories via kinesthetic guiding. CB-i is,

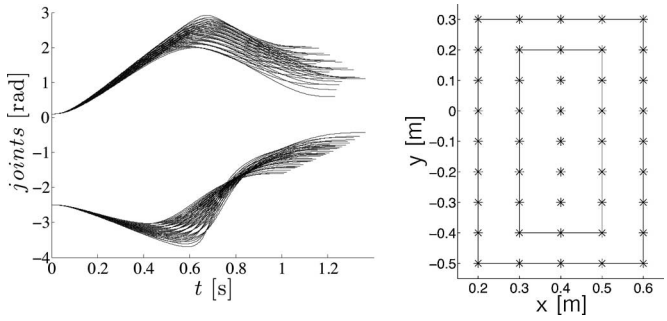


Fig. 3. Forty-five example trajectories in joint space calculated via inverse kinematics from Cartesian space minimum-jerk trajectories (left) and the final reaching points in Cartesian space (right). The Cartesian space positions are used as query points. The outer rectangle shows the full training area; the inner rectangle shows the inner training area where training data are available from all sides of the query points.

TABLE I
ERRORS IN REACHING MOVEMENTS (IN DEGREES AND CENTIMETERS, RESPECTIVELY) SYNTHESIZED BY LWR AND GPR

	Joint space (across trajectory)		Cartesian space (final position error)		Grid size (centimeters)
Training area	Full	Inner	Full	Inner	
Average error	0.24	0.19	0.12	0.09	10×10
Max. error	0.97	0.46	0.47	0.30	10×10

See Fig. 3 for the explanation of full and inner training area.

however, a much more capable system that enabled us to study the learning of control policies integrated with a full-fledged active humanoid vision. In our throwing experiments, we used a seven degree-of-freedom Mitsubishi PA-10 robotic arm.

A. Simulated Reaching Study

In the first computational study, we examined how well Cartesian minimum-jerk trajectories can be generalized. Minimum jerk trajectories are often used in robotics because they resemble human-reaching trajectories [49]. For training, we generated 45 Cartesian minimum-jerk trajectories, which were converted into joint space of a planar 2R robot (see Fig. 3, left). The final end-effector positions on the trajectories were used as query points. The training query points were distributed uniformly with spacing of 0.1 m in a rectangular area with corners at locations $(0.2, -0.5)$ and $(0.6, 0.3)$ m (Fig. 3, right). Joint velocities and accelerations were computed analytically.

As described in Section II, we applied LWR to synthesize the form parameters $\mathbf{w}$ of the DMP, while GPR was utilized to learn the function from query points (desired goal position in Cartesian space) to attractor points (the final joint configuration) and to time duration. Thus, here the relationship between query points and attractor points is given by the standard inverse kinematics. While problems with nonconvexity of the movement space[2] may arise in less-constraint situations, we did not observe such problems in our experiments. This observation is probably due to the fact that we do not need to learn a full inverse kinematics of the robot. Hence, the inverse kinematics learning is limited to a small portion of the workspace covered by query points, and the learning procedure is less likely to encounter the potentially problematic areas.

[2]These problems can be addressed by biasing [50] or re-weighting [51] the data.

TABLE II
ERRORS IN REACHING MOVEMENTS (IN DEGREES AND CENTIMETERS, RESPECTIVELY) GENERATED BY A SINGLE DMP, WHICH WAS TRAINED TO REPRODUCE ONE OF THE EXAMPLE TRAJECTORIES

	Joint space (across trajectory)		Cartesian space (final position error)	
Training area	Full	Inner	Full	Inner
Average error	8.85	5.62	0.43	0.32
Max. error	22.47	13.88	0.97	0.77

Only the attractor point was modulated, whereas the other DMP parameters were kept constant. See Fig. 3 for the explanation of full and inner training area.

The errors in Tables I and II were computed by comparing generalized joint trajectories $\tilde{\mathbf{y}}(t_j)$ (calculated by integrating (24)–(26), shown later, with the analytically computed minimum-jerk trajectories $\mathbf{y}(t_j)$, both specified in the robot joint space. The average (18), shown below, and maximum error (19), shown below, on the trajectories were estimated as

$$\text{error}_{\text{average}} = \frac{1}{T}\sum_{j=1}^{T} \|\tilde{\mathbf{y}}(t_j) - \mathbf{y}(t_j)\| \tag{18}$$

$$\text{error}_{\max} = \max_{j=1,\ldots,T} \|\tilde{\mathbf{y}}(t_j) - \mathbf{y}(t_j)\|. \tag{19}$$

Query points, at which the generalized and analytical trajectories were computed, were sampled across the complete training space. Results in Table I demonstrate that minimum-jerk movements can be generalized with high precision. The generalized trajectories accurately approximate the spatial course of movement and the final configuration. Since it can be expected that errors are larger on the boundary of query points used for training, we estimated the errors both within the full rectangular area enclosed by all query points of Fig. 3 and in a more limited inner training area enclosed by query points situated at least one query point away from the boundary points. As expected, the errors were significantly smaller when testing was limited to the internal training area.

Table II shows that representation with only one DMP is too rough for a precise movement reproduction. While the final position could be reached accurately due to the properties of DMPs, the trajectory-reproduction accuracy (18) is worse by an order of magnitude compared with the precision of the proposed approach. The columns describing the Cartesian-space error in both tables show that GPR is successful at estimating the inverse kinematics of the robot in a limited subset of the workspace. The error is larger in Table II than in Table I because the execution was stopped at time τ. The DMP obtained by generalization arrives at the goal within the specified time, whereas the DMP that was trained to reproduce one of the example trajectories needs more time to reach a modified attractor point.

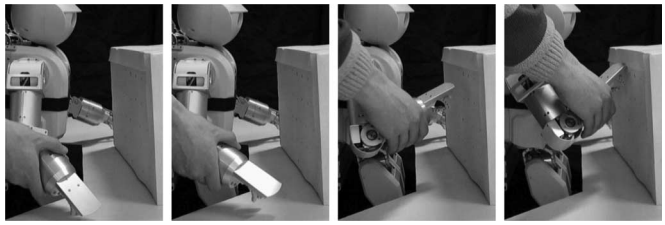


Fig. 4. Image sequence showing the teaching of reaching trajectories to HOAP-3 humanoid robot with kinesthetic guiding. Teaching of reaching movements to CB-i was done in a similar way.

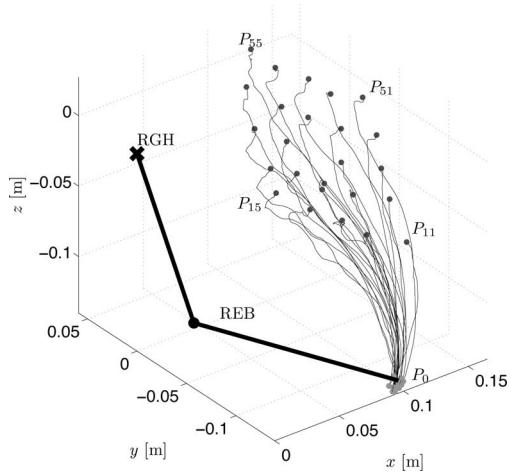


Fig. 5. Query points and example trajectories for the reaching experiment on HOAP-3. The goals, i.e., final destinations, of the demonstrated movements are marked with circles. The robot's arm with the shoulder (RGH) and elbow (REB) joints is also shown.

B. Real-World Reaching

We also tested the proposed approach for the generalization of reaching movements on a humanoid robot HOAP-3. By means of kinesthetic guiding (see Fig. 4), we recorded a set of 25 reaching movements, which were used to build a library of example movements. The example trajectories in task space and the associated query points are presented in Fig. 5. All movements roughly originated from the same starting position (P_0) and ended on a grid roughly in the coronal plane of the robot, ~ 0.15 m in front of it.

The robot's forward kinematics was used to obtain the final end-effector's positions in Cartesian space on the trajectory. Gaussian process regression (see Section II-D) was applied to estimate the function from these positions, which served as query points, to the corresponding joint angles at the final configuration on the trajectory. Since the robot is redundant with respect to the task, GPR also learns to resolve redundancies within the demonstrated, limited subset of the workspace using the same redundancy resolution strategy as demonstrated during training. Four joints (three in the shoulder and one in the elbow) were used in this experiment.

Fig. 6 shows the generalized Cartesian trajectories projected onto the yz plane. The generalized trajectories, which are presented in bold, resemble the example reaching trajectories associated with the nearest demonstrations, which are presented by thin solid lines. The generalization procedure continuously transitions between example trajectories when changing the query

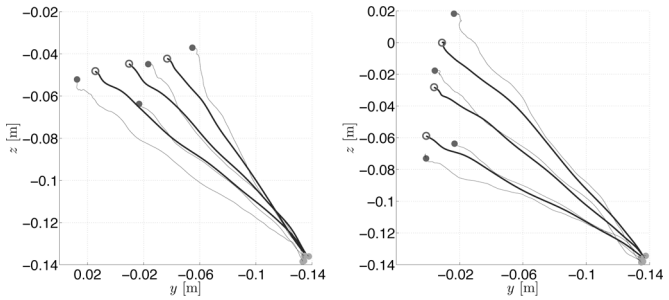


Fig. 6. Evaluation of a real-world reaching experiment. Both graphs show the results of generalization when query points smoothly change along a dimension of the query-point space. The generalized joint trajectories (bold) resemble the trajectories of the nearby demonstrations (thin). The left graph shows the changing of the query point in y direction and the right graph in the z direction. Note that the y axis is reversed to preserve similarity to Fig. 5. Query points associated with training trajectories are presented with full circles and the query points for generalized movements with empty circles.

points in both the y and z directions, while x remains roughly the same, just as in the demonstrations. Thus, we can confirm the simulation results from Section III-A in a real-world experiment.

C. Integration With Active Vision and Grasping

In our next experiment, we focused on the integration of the proposed approach with active vision and grasping, which demonstrates high performance that can be achieved by the developed system (see Figs. 7 and 8). The role of active vision is to provide parameters characterizing the task, i.e., query points. In the case of reaching and grasping, the task is characterized by the position of an object to be grasped. The use of active vision is essential if the robot is to find and grasp objects in a natural way. In this experiment, we used a full-size humanoid robot CB-i. Like in the case of HOAP-3, the training data were obtained by guiding the robot through 25 example trajectories. Besides reaching toward the desired positions, the acquired trajectories also avoid the table. Unlike in the case of HOAP-3, the positions of the target object were acquired by the robot's own visual system.

CB-i's oculomotor system has seven degrees of freedom (three in the neck and two in each eye), which ensures flexibility when the robot needs to find and direct its view toward new objects. To compute 3-D data in body coordinates by stereo vision, a robot with such a visual system must continuously update the position and orientation of the cameras, which depend on the current joint configuration of the robot, in the robot-body coordinate frame. The appropriate estimation and calibration procedures are described in [52]. This work showed that even with carefully designed calibration procedures, it is difficult to estimate all necessary coordinate-frame transformations with high accuracy. As a consequence, we cannot rely on a very accurate 3-D vision on a full-size humanoid robot with a high-degree-of-freedom oculomotor system.

The core part that integrates active vision with the developed generalization approach is the estimation of the function that maps the desired Cartesian positions (query points) to the final joint positions (DMP-attractor points) on the training

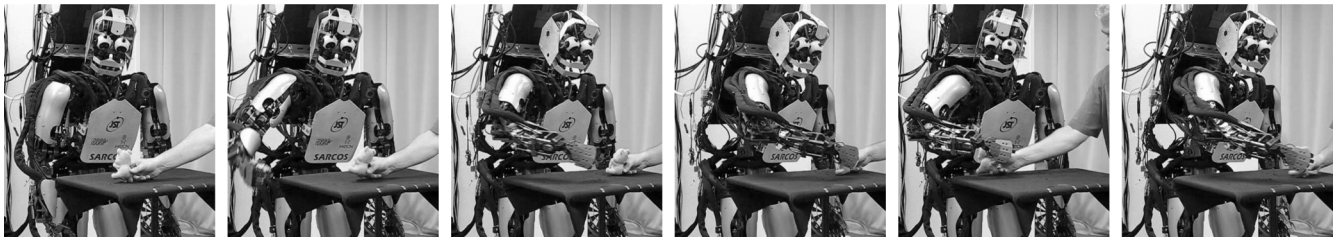


Fig. 7. Image sequence showing the robot reaching over the edge of the table toward a target. The robot tracks the target with its own eyes and determines the target's location, which is used to generate a new attractor point g for the DMP using GPR. This process runs in real time. The initially open-loop movement that avoids the table thus gradually transforms into a closed-loop movement that follows the object. Note the different head and eyes postures in the images.

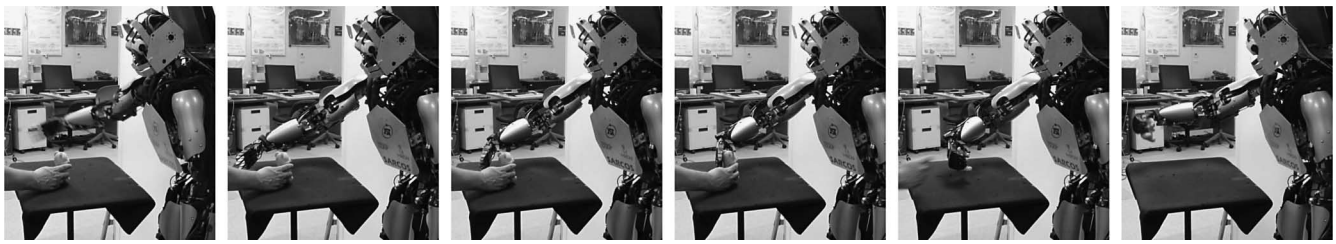


Fig. 8. Image sequence illustrating the grasping process. The head and eyes actively follow the object. The robot applied a power grasp to pick up the object.

trajectories. This function is estimated by Gaussian process regression as outlined in Section II-D.

As shown in Fig. 7, the robot needs to avoid the table while reaching toward the object. Based on the training trajectories that avoid the table, the system generates new trajectories that also avoid the table, but at the same time end at the desired Cartesian positions. Without generalizing the shape parameters $\mathbf{w}$, the robot's arm would collide with the table. The first three images of Fig. 7 and video **discrete-reach-open.mov**, which is attached to this paper and can be downloaded from http://ieeexplore.ieee.org, show the successful open-loop reaching movements that avoid the table. Once the robot's arm reaches over the table, we start updating the attractor point $\mathbf{g}$ in real time using the results of active vision and GPR. No additional programming was needed to create the closed-loop behavior, just active vision was allowed to update the attractor points. Theoretically, it is possible to update the shape parameters $\mathbf{w}$ as well, but depending on the amount of data, this calculation may require a few hundred milliseconds and is therefore unsuitable for feedback control. Once the time exceeds τ, the influence of parameters $\mathbf{w}$ starts diminishing and the robot generates its motion based solely on the attractor dynamics and GPR, which again results in collision-free movements (because the training examples are collision-free). The last three images of Fig. 7 and the attached video **discrete-reach-closed.mov** show that the robot can successfully follow a moving object, thus demonstrating the power of the DMP representation when reacting to the external feedback and the accuracy of GPR when filtering visual information.

The resulting robot-hand path while following the object in a closed-loop is depicted in Fig. 9. The mean value of the robot-hand position while following the object that moves on the table (as calculated by forward kinematics) and the mean value of the object position (as estimated by active vision) differ by about $[1.6, 4.2, 7.6]$ cm. Although the real modeling errors are nonlinear, the error would be roughly similar if we just estimated object positions and converted them into joint angles by means of inverse kinematics. Such an error would be too large for reaching and grasping. In our system, GPR successfully corrects at least part of the modeling errors and the resulting attractor points are accurate enough for these tasks. This is due to the fact that already during training, the system directly relates the object positions as estimated by vision to the final robot-joint configurations. The vision errors can therefore be taken into account by GPR, the system only needs to be repeatable.

The realized grasping behavior is shown in Fig. 8 and in the attached video **discrete-grasping.mov**. Active vision detects when the object stops moving and initiates the open-loop reaching. The attractor points for the initial reaching trajectory are generated by suitably displacing the estimated query points (based on the desired approach direction). After the initial reaching movement has finished, the vision system starts supplying the current object position and the second-order attractor dynamics automatically generates a closed-loop approach motion for grasping. The system is accurate enough to pick an object from the hand of a person. Note that without active eye and head joints, the robot would not be able to follow the object, especially when it comes close to the body, and the behavior could not be generated.

D. Simulated Ball Throwing

To demonstrate the performance of the approach for the synthesis of more dynamic tasks, we considered the task of throwing a ball into a basket using the same 2R robot for simulation as in reaching. Throwing depends not only on the positional part of the movement but on velocities as well. The trajectory of the ball after the release is fully specified by the position and

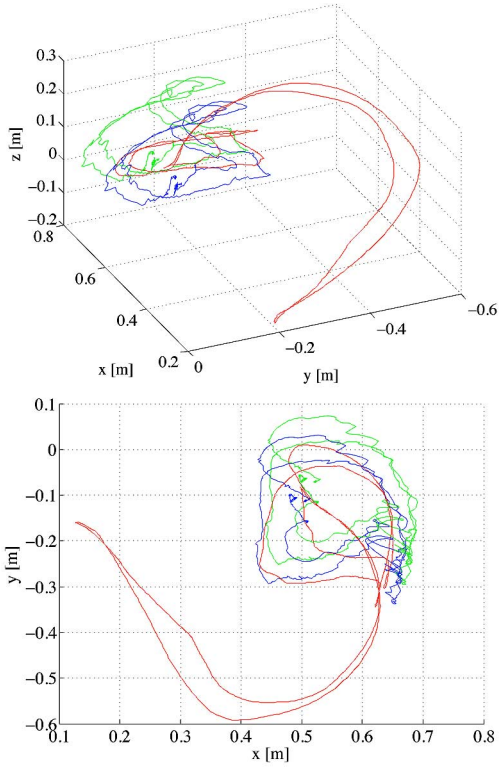


Fig. 9. The upper image shows the 3-D path of the robot's hand (calculated by forward kinematics) while reaching and tracking an object (red). The object positions estimated by active vision are shown in green, while the blue curve shows the estimated object positions with corrected mean value so that it coincides with the mean of the hand motion. The lower image shows the same data projected onto a plane, which is roughly parallel to the table. The proposed approach is successful at correcting systematic vision errors.

velocity at the release time

$$x = x_0 + v_0 t\cos(\alpha),\ y = y_0 + v_0 t\sin(\alpha) - \frac{gt^2}{2} \qquad (20)$$

where (x_0, y_0) is the release point, v_0 is the linear velocity of the ball at the release time, and α is the initial angle of the throw path. We considered the problem where the target basket is placed in xy-plane. The understanding of the physics of the task allows us to compare movement generalization results with an ideal system. Local models were also studied in [53] to refine the throwing performance but not to generalize the throws over the complete training space.

The target positions, i e., the positions where the ball is supposed to land, were used as query points. The training-query points were uniformly distributed within a rectangular area with corners at $(1.2, 0.1)$ and $(5.2, 2.1)$ m, with different spacings as shown in Table III. Based on (20), example movements with proper position and velocity at release times were analytically generated for the training targets. To avoid accurately modeling the physical grip and the release of the ball from the hand—which is rather complex to model but would not contribute much to our analysis—we also attached the release time to every training trajectory (besides the usual duration and attractor points). This was not necessary to do on a real robot (see Section III-E). For evaluation, new query points were generated

TABLE III
ERRORS IN THE SYNTHESIZED BALL THROWS (DISTANCE FROM THE TARGET IN CENTIMETERS)

	Adams-Bashforth-Moulton integration		Euler integration		Grid size (centimeters)
Training area	Full	Inner	Full	Inner	
Average error	2.66	2.00	4.58	3.64	50×50
Max. error	16.5	6.87	22.78	7.21	50×50
Average error	1.19	0.41	3.91	3.36	25×25
Max. error	14.15	2.83	20.55	6.34	25×25
Average error	0.35	0.13	3.28	3.10	12.5×12.5
Max. error	6.05	0.44	13.48	6.11	12.5×12.5
Average error	0.19	0.11	3.13	3.04	6.25×6.25
Max. error	2.83	0.27	9.78	6.09	6.25×6.25

The throws were sampled in the area of 4×2 meters. The number of example trajectories was 45, 153, 561, and 2145 for different grid sizes, respectively.

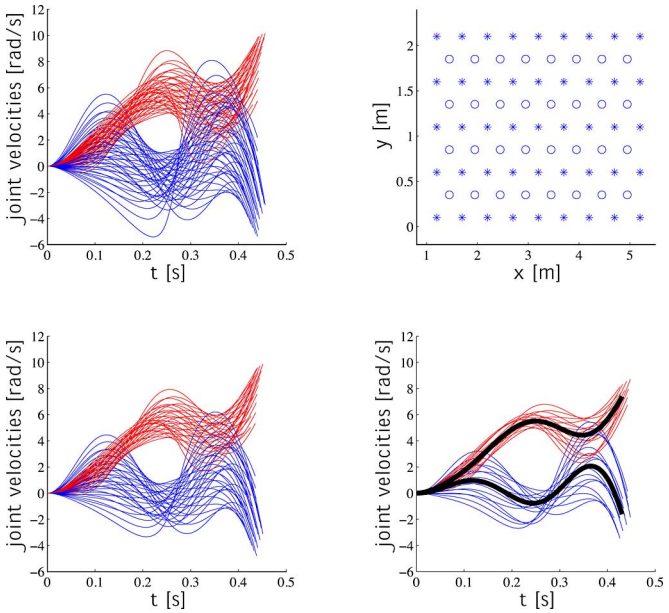


Fig. 10. Simulation results. (Top left) Joint velocities originating from 45 training trajectories. Only the relevant parts of the throwing trajectories (up to the release time) are displayed. (Bottom left) Generalized velocities for 32 in-between query points. They end at the release times estimated by GPR. (Top right) Training target positions (stars) and the in-between query points (circles). (Bottom right) The generalized velocities (thick black lines) for query point $\mathbf{q} = [2.95, 1.35]$ and the velocities of nearby training trajectories that were used for generalization. Our approach preserves the trends.

on a finer grid than for training (2 cm), and the corresponding DMPs were calculated using the proposed generalization approach. The release times were also estimated by GPR. During execution, the ball was detached from the hand at the estimated release time. Using the position and velocity at release time, we could calculate where the ball would land using (20).

Fig. 10 depicts the velocities of the most coarsely sampled training movements and their generalization. It can be seen that the generalized velocities and release times are similar to the training velocities and release times. This is confirmed in

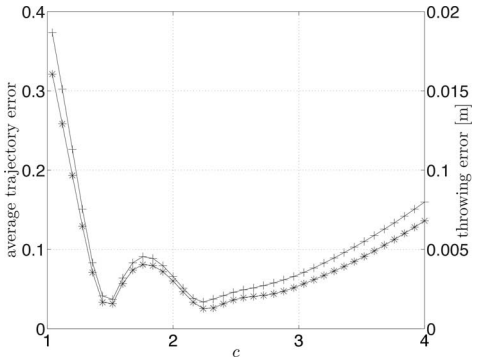


Fig. 11. Validation-set error as a function of the bandwidth parameter c defined in (15). The curve with stars shows the average difference (18) between the synthesized trajectories and validation trajectories, and the curve with pluses shows the error of the synthesized ball throws. They have similar minima.

Table III, which shows that we can generate throws to any target within the training space with high accuracy. Even when the training targets were distributed with the spacing of 0.5 m, we could achieve the average accuracy of 2.0 cm for the inner training area where a sufficient number of example movements are available. The inner training area was defined similarly as in Fig. 3, with the width of the border area being equal to the spacing between the example query points. The error dropped to 0.13 cm for training targets distributed with the spacing of 12.5 cm. Thus, we can say that our approach produces throws with respectable accuracy, even when the training examples are sparse, and that the accuracy improves with more densely distributed training data. These results also show that the accuracy is higher when a more advanced integration technique such as Adams–Bashforth–Moulton method [54] is used to integrate the system (24)–(26), shown later. Errors were significantly larger when we applied the simpler Euler's method. While the generalized DMPs approximate the desired trajectories and their dynamics with high precision, care must be taken when integrating the generalized DMPs to reproduce this accuracy, which is important in tasks such as ball throwing.

We also tested the automatic calculation of the bandwidth parameter c of (15) for the locally weighted regression using the validation-set-error method. As can be seen in Fig. 11, the criterion has two local minima. The accuracy of the results was slightly better with the larger c, but similar accuracy and faster computational times could be achieved with the smaller value of c. Note that the minima are very similar if we evaluate the reproduction accuracy of the synthesized motion or if we measure the accuracy of throwing. This fact is important because the former does not require new experiments with the robot, whereas the latter can only be calculated if the robot performs new throws. This makes the second approach impractical for real-world training.

E. Real-World Ball Throwing

To test real throwing, we recorded 20 throwing trajectories, which were manually trained for different targets, and measured where the ball landed for each of these trajectories. We used three of the total seven degrees of freedom of PA-10 robot to realize robotic throwing. A coaching process similar to the one described in [55] was applied to accumulate the training data. Note that instead of using the actually executed movements as training data for generalization, it was better to use the commanded trajectories. In this way, any discrepancies between the commanded and actually executed trajectories can be taken into account by the generalization process. In addition, the commanded trajectories are smoother. Unlike in simulations, here, we did not need to consider the release times because the release times are implicitly encoded in the arm trajectories. In our experiments, the ball was held only loosely by the gripper. It

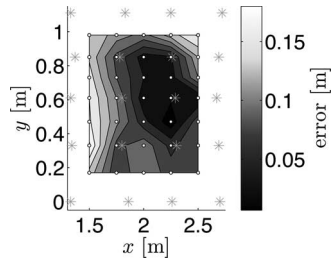


Fig. 12. Accuracy of real throwing. The stars are the training targets while the dots show the input query points for generalization. For evaluation purposes, the robot executed three throws at each target, and we calculated the average throwing error from the desired target (query point).

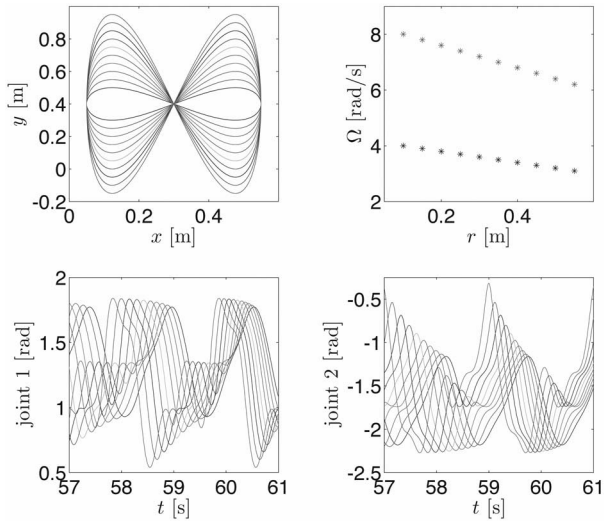


Fig. 13. (Top left) Library of example Fig. 8 trajectories in Cartesian space. (Top right) Frequencies of example movements in x and y direction. (Bottom left) Trajectories of joint 1 of the 2R planar robot. (Bottom right) Trajectories of joint 2 of the 2R planar robot.

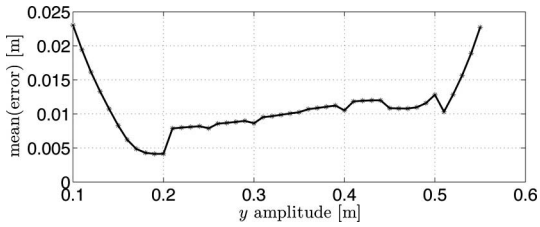


Fig. 14. Mean error of movement generalization over the entire scope of the library of example movements estimated with the step of 1 cm.

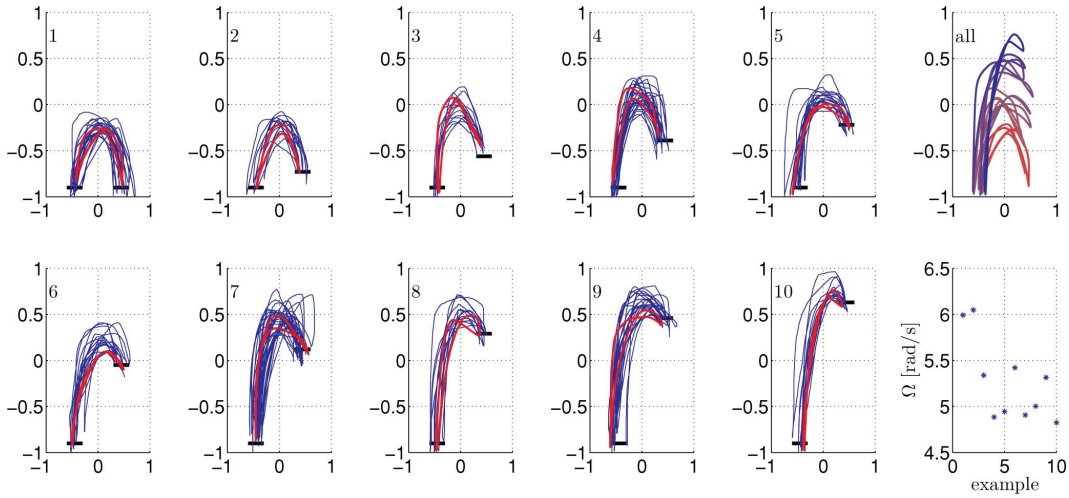


Fig. 15. Example periodic trajectories in Cartesian space. The blue curves show the training input, and the red curves show the estimated periodic trajectories. To transform the demonstrated screen trajectories into Cartesian space, the values have to be multiplied by 0.2 m. All final estimated training trajectories are in the same plot in the top-right image. The frequencies are in the bottom-right plot.

was detached from the hand due to the dynamics of motion. The repeatability of throws was about 2–3 cm; hence, this is the best accuracy achievable by learning. Fig. 12 shows that we can obtain accuracy of about 2–10 cm for the throws within the inner training area, which is quite good when we consider the repeatability of the robot and that the spacing between training throws was about 0.5 m. The accuracy needed to get the ball into the basket was within few centimeters. As expected, the robot performed worse for the targets at the edge of the training area, where less training data are available, and the precision is lower (see Fig. 12 and the attached video **discrete-throwing.mov**, in which the robot misses the last three throws into a basket placed at the edge of the training area).

F. Simulated Periodic Pattern Study

Similarly to our analysis, in the case of discrete movements, we conducted a simulation experiment to examine how well we can generalize periodic Cartesian trajectories of a robot end-effector. Just like in the simulated reaching study, we used a planar 2R robot for simulation.

As an example periodic motion, we chose a Cartesian figure-8 trajectory with a varying amplitude and frequency. Several reasons speak for this, namely we can analytically generate a signal with an arbitrary amplitude for comparison and the motion frequencies of the two separate dimensions in Cartesian space are different. While the primary aim of the generalization algorithm is to produce DMPs for movements that cannot be attained by simple modulation, for the purpose of evaluation, this section uses an example that can be easily attained by modulating the amplitude-control parameter of the periodic DMP. The next section gives an example where generalization could not be achieved by simple modulation.

A set of example figure-8s in Cartesian space and the trajectories in joint space is depicted in Fig. 13. Cartesian-space trajectories are distributed evenly from the smallest trajectory at $x = 0.3 + 0.2\cos(\Omega t)$ m and $y = 0.4 + 0.1\sin(2\Omega t)$ m to the largest at $y = 0.4 + 0.55\sin(2\Omega t)$ m with a step of 5 cm (x does not change). Ten sampled joint-space movements are used to form the example library.

Using the amplitude in the y dimension as a query-point parameter, we generalized the training-joint trajectories, i.e., we calculated the parameters of a periodic DMP as described in Section II-C and D. Different frequencies in each dimension of the example demonstration movements (see Fig. 13) show that we can successfully estimate the periodic movements onto a single period. Once the new movement is generated, frequencies can easily be modulated with the periodic DMP frequency-control parameter, just like it is done in a standard DMP approach (see [44] for details).

Fig. 14 shows the mean error of movement generalization over the entire scope of the example library. As before, the distance of the generalized trajectory from the analytically computed trajectory increases at the edge of the database as there, example movements are available only on one side of the query point. The accuracy of generalization is slightly decreasing, which is the result of the increasing amplitude.

G. Real-World Drumming Experiment

In the final experiment, we applied the proposed approach to realize drumming on humanoid robot CB-i. Of the total 39 degrees of freedom, we used all seven of the right arm for the execution of the movement. To demonstrate generalization, we trained the robot to perform one-handed drumming on a drum and a cymbal, where the cymbal was mounted at different heights. Such drum placements—although with more drums—are common for drummers. Fig. 15 and the attached video **periodic-examples.mov** show a set of example periodic trajectories in Cartesian space, which were recorded with a sampling rate of 100 Hz. The trajectories were demonstrated with a mouse on a screen, scaled to Cartesian space, and mapped onto the robot's joint angles via inverse kinematics in real time. The trainer was modifying his motion

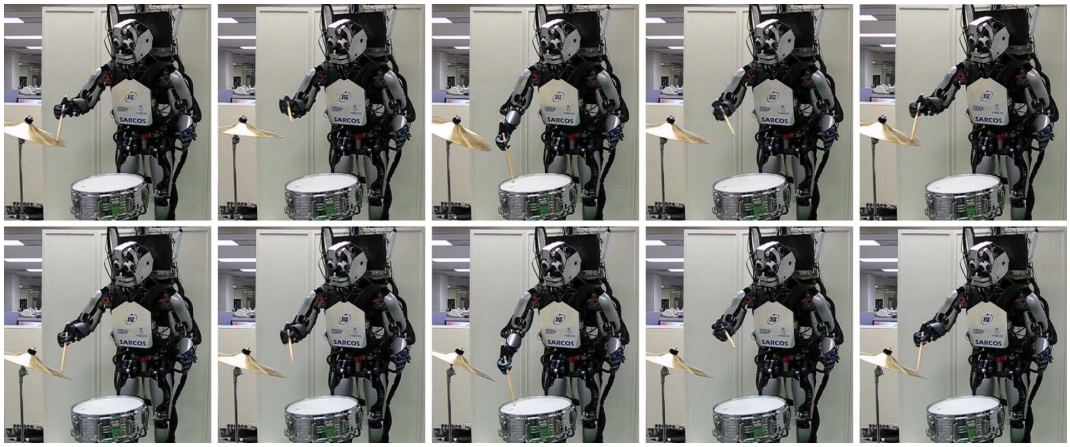


Fig. 16. Image sequence showing the drumming at two different heights. The cymbal is higher in the top row.

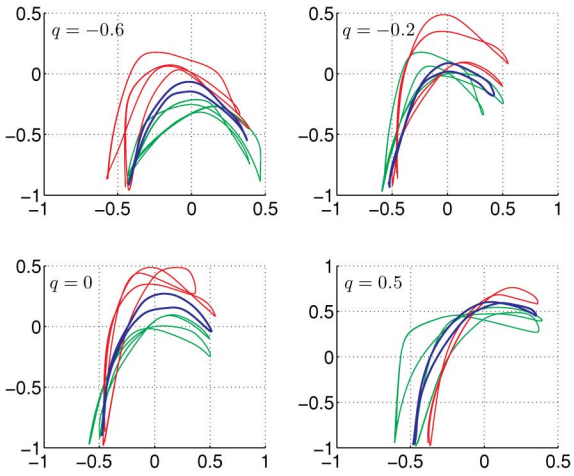


Fig. 17. The result of generalization for four randomly chosen query points. The figure shows the generalized trajectories in blue. The training trajectories associated with query points just above (red) and below (green) the current query point are also shown. Among the ten example trajectories, these trajectories influenced the generalization process most.

based on the real-time visual feedback coming from the robot. The drum was placed at -0.162 m. For training, we mounted the cymbal at ten different heights ($-0.177, -0.149, -0.091, -0.068, -0.049 - 0.020, 0.017, 0.054, 0.079$, and 0.117 m). Thus, the maximum height difference was about 30 cm.

Fig. 16 and the attached videos **periodic-generalized1.mov** and **periodic-generalized2.mov** show the ability of the system to generalize the trained periodic trajectories. The resulting movements cannot be attained trough simple modulation of a periodic DMP because standard modulation techniques do not allow the robot to modify the movement on only one side. The proposed approach can automatically generate the appropriate movements from the collected data. The query point for generalization is the height difference between the drum and the cymbal. While the height difference could be estimated by vision, we simplified this experiment by measuring the difference with a ruler and providing it to the algorithm.

Fig. 17 presents the results of generalization for four randomly selected query points. The figure shows the generalized periodic trajectories and the trajectories associated with query points above and below the current query point. These trajectories were used for generalization. Due to the limited support of the weighting kernel from (13), the generalized trajectories are generated only from the four plotted example trajectories. As the trajectories were generated by human demonstrations, they are quite different among themselves. Nevertheless, the algorithm was able to generate trajectories similar to the example trajectories. If the demonstrations were more uniform, the generalized trajectories would be as well.

IV. Final Discussion and Conclusion

Our experiments show that the proposed approach is able to generalize the training data to new situations within the sampled training space. The computational time required for generalization compared to the computational time for standard one-shot learning of DMPs increases only linearly with a number of training trajectories that are taken into account by local weighting. The relatively low computational complexity of the method enabled us to compute new control policies directly from the sampled data online. In this way, we avoid the pitfalls associated with the projection of the training data into lower dimensional parameter spaces (also called latent spaces), which can lead to oversmoothing and, therefore, losing important details of the task.

In contrast to reinforcement learning (RL), where a robot actively explores the solution space to adapt its movements to new situations, we focused on generalization to new situations from the available data. The two methodologies are complementary; RL techniques can be used to provide initial samples for our trajectory libraries. On the other hand, our method can be used to provide a good initial approximation for RL (thus speeding up the convergence) if higher accuracy than what can be achieved by the proposed approach is required.

Our method is appropriate if the example trajectories smoothly transition as a function of query points, as defined in (2). Otherwise, nearby data do not provide information about the movement associated with the new query point $\mathbf{q}$. A more complex relationship might require a larger number of training trajectories. It is, however, intuitive that this relationship is smooth in real-world problems because it is unlikely that a totally

different strategy would be used to solve a task in different but similar situations. Instead, the movement trajectories are usually adjusted when solving the task at nearby query points. Our experiments show that even though the available training sets were not large in real-world test, we could successfully generalize and achieve high performance in several tasks that are relevant for robotics.

The proposed generalization process synthesizes DMPs in a standard form, and thus, we can still exploit all of the advantages of standard DMPs at execution time, like, for example, modulation of the time evolution of the phase by modifying (24) [20], avoiding the joint limits by modifying (22) [44], etc. We have also demonstrated in Section III-C that the developed system can be used in an active feedback loop by modifying the attractor point of a DMP online, with GPR correcting some of the modeling errors causing uncertainties in the 3-D vision data.

There exist tasks in which example movements do not transition smoothly as a function of query points. Consider, for example, reaching movements that need to avoid an obstacle before arriving at the final destination. If there are two sets of example movements, each avoiding the obstacle from a different side, then example movements that avoid the obstacle from different sides should not be used for generalization simultaneously. The proposed approach could still be used, but it would need to be supplemented by a suitable clustering procedure that determines sets of trajectories suitable for generalization. Issues like clustering of example movements and the identification of motor primitives are important for cognitive robots that should gradually increase their competence through seamless learning. While we did not consider such issues in this paper, our approach provides a methodology to generalize the appropriately selected training trajectories in a natural way, thus providing an important building block for a cognitive robot.

APPENDIX A
CONTROL POLICIES AS DYNAMIC SYSTEMS

Here, we briefly explain the theoretical fundamentals of the motor representation developed by Ijspeert *et al.* [28], [29]. These authors proposed to describe a control policy by a set of nonlinear differential equations with a well-defined attractor dynamics. Here the most current formulation as outlined in [20] is used. For a single degree of freedom denoted by y, which can either be one of the internal joint angles or one of the external task-space coordinates, the following system of linear differential equations with constant coefficients has been proposed as a basis for motion specification:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) \tag{21}$$

$$\tau \dot{y} = z. \tag{22}$$

Provided that the parameters $\alpha_z,\ \beta_z$, and $\tau > 0$ are selected appropriately, e.g., $\alpha_z = 4\beta_z$, this system has a unique attractor point at $y = g,\ z = 0$.

Differential equations (21)–(22) ensure that y converges to g and can therefore be used to realize discrete point-to-point movements. To increase a rather limited set of trajectories that can be encoded by (21) and (22) and thus enable the approximation of general point-to-point movements, (21) needs to be modified. In the case of discrete movements, one can add a linear combination of radial-basis functions to (21) [20][3]

$$f(x) = \frac{\sum_{i=1}^{N} w_i \Psi_i(x)}{\sum_{i=1}^{N} \Psi_i(x)} x,\ \Psi_i(x) = \exp\left(-h_i\,(x - c_i)^2\right) \tag{23}$$

where c_i are the centers of radial basis function distributed along the trajectory, and $h_i > 0$. A phase variable x is used in (23) instead of time to make the dependency of f on time more implicit. Its dynamics can be defined by

$$\tau \dot{x} = -\alpha_x x \tag{24}$$

with initial value $x(0) = 1$. A solution to (24) is given by $\exp(-\alpha_x t/\tau)$, thus x tends to 0 as time increases. As shown in [20], the appealing property of using the phase variable x instead of explicit time is that by appropriately modifying (24), we can, for example, stop the evolution of time to account for perturbations during trajectory execution. This results in the following system of differential equations:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x) \tag{25}$$

$$\tau \dot{y} = z \tag{26}$$

which can be used to approximate discrete movements of various shapes. Since x tends to zero, the influence of the nonlinear term $f(x)$ decreases with time and system (25) and (26) converges to $[0, g]^T$, just like the system (21) and (22). The control policy specified by variable y defines what is called DMP.

In the case of periodic movements, the following linear combination of periodic functions can be used to change the dynamics of the basic second-order system [20]:

$$f(\phi) = \frac{\sum_{i=1}^{N} w_i \Gamma_i(\phi)}{\sum_{i=1}^{N} \Gamma_i(\phi)} r,\ \Gamma_i(\phi) = \exp\left(h_i\,(\cos(\phi - c_i) - 1)\right) \tag{27}$$

where r is the amplitude of the oscillator, and $h_i > 0$. Writing $\tau = 1/\Omega$, (25) and (26) are replaced by

$$\dot{z} = \Omega\,(\alpha_z(\beta_z(g - y) - z) + f(\phi)) \tag{28}$$

$$\dot{y} = \Omega z. \tag{29}$$

The phase variable ϕ has been introduced in this case to avoid the explicit dependency on time. The phase is assumed to move with constant speed

$$\dot{\phi} = \Omega \tag{30}$$

where Ω is the frequency of oscillation.

DMPs have been designed to provide a representation that enables accurate encoding of the desired trajectories and at the same time permit the modulation of different properties of the encoded trajectories. In this context, the shape parameters w_i

[3] f defined in [20] is scaled by $g - y_0$, i.e., $f(x) = [\sum_{i=1}^{N} w_i \Psi_i(x) / \sum_{i=1}^{N} \Psi_i(x)] x(g - y_0),\ y_0 = y(0)$. Thus, when the attractor point g changes, the encoded movement gets scaled. We omit this scaling factor because we are not interested in automatic scaling, which is achieved differently in our approach. If g is kept constant, the scaling factor has no effect.

are determined so that the robot can accurately follow the desired trajectory by integrating (24)–(26) or (28)–(30). The other parameters are used for modulation. For example, by changing g, we can adjust the final destination of a discrete movement, while τ can be adjusted to modulate its velocity. Similarly, Ω can be changed to modulate the frequency of oscillation in case of periodic movements.

APPENDIX B
ADAPTIVE FREQUENCY OSCILLATORS

The frequency of oscillation is not directly observable in the data. To estimate the frequency, Righetti *et al.* [43] suggested to replace the constant-speed assumption (30) by a system

$$\dot{\phi}_i = \Omega_i - Ke\,(t)\sin\,(\phi_i) \tag{31}$$

$$\dot{\Omega}_i = -Ke\,(t)\sin\,(\phi_i) \tag{32}$$

$$\dot{\alpha}_i = \eta\cos\,(\phi_i)\,e\,(t) \tag{33}$$

where $e\,(t) = y_{\rm d}\,(t) - \hat{y}(t)$, and $\hat{y}(t) = \sum_{i=1}^{L}\alpha_i\cos\,(\phi_i)$. Note that if $e\,(t) = 0$, the system (31)–(33) becomes equivalent to (30). It has been shown that by integrating this system, the frequencies Ω_i contained in the observed motion trajectory can be estimated. The most significant frequency is selected as the base or fundamental frequency Ω for the DMP (see [44] for the integration of adaptive frequency oscillators with DMPs).

APPENDIX C
GAUSSIAN PROCESS REGRESSION

A Gaussian process is defined as

$$g(\mathbf{q}) \sim \mathcal{G}P\,(m(\mathbf{q}), k(\mathbf{q}, \mathbf{q}')) \tag{34}$$

where $m(\mathbf{q}) = \mathbb{E}(g(\mathbf{q}))$ is the mean function, and $k(\mathbf{q}, \mathbf{q}') = \mathbb{E}((g(\mathbf{q}) - m(\mathbf{q}))(g(\mathbf{q}') - m(\mathbf{q}')))$ is the covariance function of the process. Let us assume that we have a set of noisy observations $\{(\mathbf{q}_k, y_k)|k = 1, \ldots, M\}, y_k = g(\mathbf{q}_k) + \epsilon,\ \ \epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Subtracting the mean from the training data, we can further assume that $m(\mathbf{q}) = 0$. If we are given a set of query points $g(\mathbf{q}^*)$, then the joint distribution of all outputs is given as

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I} & \mathbf{K}(\mathbf{Q}, \mathbf{Q}^*) \\ \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}) & \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}^*) \end{bmatrix}\right) \tag{35}$$

where $\mathbf{Q}$, $\mathbf{Q}^*$, $\mathbf{y}$, $\mathbf{y}^*$, respectively, combine all inputs and outputs, and $\mathbf{K}(\cdot, \cdot)$ are the associated joint covariance matrices calculated according to (34). It can be shown [42] that the expected value $\bar{\mathbf{y}}^*$ is given by

$$\bar{\mathbf{y}}^* = \mathbb{E}(\mathbf{y}^*|\mathbf{Q}, \mathbf{y}, \mathbf{Q}^*) = \mathbf{K}(\mathbf{Q}^*, \mathbf{Q})[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y} \tag{36}$$

with the following estimate for the covariance of the prediction:

$$\begin{aligned}\operatorname{cov}(\mathbf{g}^*) = {} & \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}^*) \\ & - \mathbf{K}(\mathbf{Q}^*, \mathbf{Q})[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2\mathbf{I}]^{-1}\mathbf{K}(\mathbf{Q}, \mathbf{Q}^*).\end{aligned}$$

One commonly used covariance function is

$$k(\mathbf{q}, \mathbf{q}') = \sigma_f^2\sum_{i=1}^{n}\exp\left(-\frac{1}{2}\frac{(q_i - q_i')^2}{l_i^2}\right) \tag{37}$$

which results in a Bayesian regression model with an infinite number of basis functions. Here, n denotes the dimension of the query-point space. See [42] for more details.

REFERENCES

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[2] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robot. Auton. Syst.*, vol. 47, no. 2–3, pp. 109–116, 2004.

[3] M. Riley, A. Ude, and C. G. Atkeson, "Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching," in *Proc. 2000 Workshop Interactive Robot. Entertainment*, Pittsburgh, PA, 2000, pp. 35–42.

[4] N. S. Pollard, J. K. Hodgins, M. Riley, and C. G. Atkeson, "Adapting human motion for the control of a humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, 2002, pp. 1390–1397.

[5] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robot. Auton. Syst.*, vol. 47, no. 2–3, pp. 93–108, 2004.

[6] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Humanoid robot motion generation with sequential physical constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, 2006, pp. 2649–2654.

[7] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Comput. Vis. Image Understanding*, vol. 104, no. 2, pp. 90–126, 2006.

[8] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1463–1467, Dec. 2008.

[9] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, pp. 682–697, 2008.

[10] M. Mistry, P. Mohajerian, and S. Schaal, "An exoskeleton robot for human arm movement study," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Edmonton, Canada, 2005, pp. 4071–4076.

[11] A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A unified passivity based control framework for position, torque and impedance control of flexible joint robots," *Int. J. Robot. Res.*, vol. 26, no. 1, pp. 23–39, 2007.

[12] S.-H. Hyon, J. G. Hale, and G. Cheng, "Full-body compliant human–humanoid interaction: Balancing in the presence of unknown external forces," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 884–898, Oct. 2007.

[13] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Trans. Robot. Automat.*, vol. 10, no. 6, pp. 799–822, Dec. 1994.

[14] S.-B. Kang and K. Ikeuchi, "Toward automatic robot instruction from perception – Temporal segmentation of tasks from human hand motion," *IEEE Trans. Robot. Automat.*, vol. 11, no. 5, pp. 670–681, Oct. 1995.

[15] M. Kaiser and R. Dillmann, "Building elementary robot skills from human demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, 1996, pp. 2700–2705.

[16] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Netw.*, vol. 9, no. 8, pp. 1281–1302, 1996.

[17] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Adv. Robot.*, vol. 21, pp. 1521–1544, 2007.

[18] J. Peters, J. Kober, K. Muelling, D. Nguyen-Tuong, and O. Kroemer, "Towards motor skill learning for robotics," presented at the Int. Symp. Robot. Res., Lucerne, Switzerland, 2009.

[19] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*. London, U.K.: Springer-Verlag, 2000.

[20] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control—A unifying view," *Progr. Brain Res.*, vol. 165, no. 6, pp. 425–445, 2007.

[21] T. Asfour, F. Gyarfas, P. Azad, and R. Dilmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *Int. J. Humanoid Robot.*, vol. 5, no. 2, pp. 183–202, 2008.

[22] A. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robot. Auton. Syst.*, vol. 54, pp. 370–384, 2006.

[23] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *Int. J. Robot. Res.*, vol. 23, no. 4–5, pp. 363–377, 2004.

[24] D. Kulić, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive

hidden Markov chains," *Int. J. Robot. Res.*, vol. 27, no. 7, pp. 761–784, 2008.
[25] D. Kulić, W. Takano, and Y. Nakamura, "Online segmentation and clustering from continuous observation of whole body motions," *IEEE Trans. Robot.*, vol. 25, no. 5, pp. 1158–1166, Oct. 2009.
[26] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 32, no. 2, pp. 286–298, Apr. 2007.
[27] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," presented at the Robot., Sci. Syst. Conf., Seattle, Washington, 2009.
[28] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, 2002, pp. 1398–1403.
[29] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, 2002, pp. 958–963.
[30] T. Poggio and E. Bizzi, "Generalization in vision and motor control," *Nature*, vol. 431, pp. 768–774, 2004.
[31] C. G. Atkeson, J. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, and M. Kawato, "Using humanoid robots to study human behavior," *IEEE Intell. Syst.*, vol. 15, no. 4, pp. 46–56, Jul./Aug. 2000.
[32] M. Raibert, "A model for sensorimotor control and learning," *Biol. Cybern.*, vol. 29, pp. 29–36, 1978.
[33] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning for control," *Artif. Intell. Rev.*, vol. 11, pp. 75–113, 1997.
[34] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom.*, Kobe, Japan, 2009, pp. 763–769.
[35] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Daejeon, Korea, 2008, pp. 91–98.
[36] J. Kober, B. Mohler, and J. Peters, "Learning perceptual coupling for motor primitives," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, 2008, pp. 834–839.
[37] F. A. Mussa-Ivaldi and E. Bizzi, "Motor learning through the combination of primitives," *Phil. Tran. Royal Soc. London B*, vol. 355, pp. 1755–1769, 2000.
[38] D. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Netw.*, vol. 17, pp. 1317–1329, 1998.
[39] M. Haruno, D. M. Wolpert, and M. Kawato, "MOSAIC model for sensorimotor learning and control," *Neural Comput.*, vol. 13, pp. 2201–2220, 2001.
[40] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artif. Intell. Rev.*, vol. 11, pp. 11–73, 1997.
[41] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local Gaussian process regression," *Adv. Robot.*, vol. 23, pp. 2015–2034, 2009.
[42] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.
[43] L. Righetti, J. Buchli, and A. J. Ijspeert, "Dynamic Hebbian learning in adaptive frequency oscillators," *Physica D*, vol. 216, pp. 269–281, 2006.
[44] A. Gams, A. J. Ijspeert, S. Schaal, and J. Lenarčič, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Auton. Robots*, vol. 27, no. 1, pp. 3–23, 2009.
[45] E. Oztop, J. Babič, J. G. Hale, G. Cheng, and M. Kawato, "From biologically realistic imitation to robot teaching via human motor learning," in *Neural Information Processing: 14th International Conference* (Lecture Notes in Computer Science Series). Berlin/Heidelberg, Germany: Springer-Verlag, 2008, pp. 214–222.
[46] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Comput.*, vol. 10, no. 8, pp. 2047–2084, 1998.
[47] D. W. Scott, *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: Wiley, 1992.
[48] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen, "CB: A humanoid research platform for exploring neuroscience," *Adv. Robot.*, vol. 21, no. 10, pp. 1097–1114, 2007.
[49] T. Flash and N. Hogan, "The coordination of arm movements: An experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, 1985.
[50] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Maui, HI, 2001, pp. 298–303.
[51] J. Peters and S. Schaal, "Learning to control in operational space," *Int. J. Robot. Res.*, vol. 27, no. 2, pp. 197–212, 2008.
[52] A. Ude and E. Oztop, "Active 3-D vision on a humanoid head," presented at the 14th Int. Conf. Adv. Robot., Munich, Germany, 2009.
[53] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer, "Task-level robot learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Philadelphia, PA, 1988, pp. 1309–1310.
[54] J. D. Hoffman, *Numerical Methods for Engineers and Scientists*, 2nd ed. Boca Raton, FL: CRC, 2001.
[55] M. Riley, A. Ude, C. G. Atkeson, and G. Cheng, "Coaching: An approach to efficiently and intuitively create humanoid robot behaviors," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Genoa, Italy, 2006, pp. 567–574.

Aleš Ude received the Diploma degree in applied mathematics from the University of Ljubljana, Ljubljana, Slovenia, in 1990, and the Ph.D. degree from the Faculty of Informatics, University of Karlsruhe, Karlsruhe, Germany, in 1995.

He is currently a Senior Research Associate with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He is also with Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International, Kyoto, Japan. His research interests include imitation learning, perception of human activity, humanoid robot vision, and humanoid cognition.

Dr. Ude is a recipient of the Science and Technology Agency fellowship for postdoctoral studies with the Exploratory Research for Advanced Technology (ERATO) Kawato Dynamic Brain Project, Japan.

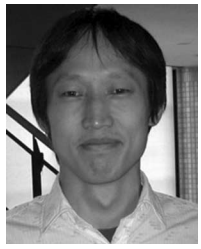
Andrej Gams received the Ph.D. degree in robotics from the University of Ljubljana, Ljubljana, Slovenia, in 2009.

He is currently a Postdoctoral Assistant with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He performed a part of his doctoral research with the Biologically Inspired Robotics Group, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland. His research interests include human arm motion and the imitation of rhythmic tasks.

Tamim Asfour received the Diploma degree in electrical engineering in 1994 and the Ph.D. degree in computer science in 2003, both from the University of Karlsruhe, Karlsruhe, Germany.

He is currently a Senior Researcher with the Karlsruhe Institute of Technology, where he is also a Leader of the Humanoid Research Group, Institute for Anthropomatics. His research interest includes humanoid robotics.

Dr. Asfour received the Forschungszentrum Informatik prize for his outstanding Ph.D. dissertation on sensorimotor control in humanoid robotics and the development of the humanoid robot ARMAR in 2003.



Jun Morimoto received the Ph.D. degree in information science from Nara Institute of Science and Technology, Nara, Japan, in 2001.

From 2001 to 2002, he was a Postdoctoral Fellow with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. Since 2002, he has been with the Advanced Telecommunications Research Institute International, Kyoto, Japan, where he was a Researcher with the Computational Brain Project, International Cooperative Research Project, Japan Science and Technology Agency, from 2004 to 2009, and is currently the Head of the Department of Brain Robot Interface, Computational Neuroscience Laboratories.