Robotics and Autonomous Systems 60 (2012) 1327–1339

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

On-line motion synthesis and adaptation using a trajectory database

Denis Forte [a,*], Andrej Gams [a], Jun Morimoto [b], Aleš Ude [a,b]

[a] Jožef Stefan Institute, Department of Automatics, Biocybernetics, and Robotics, Jamova cesta 39, 1000 Ljubljana, Slovenia

[b] ATR Computational Neuroscience Laboratories, Department of Brain Robot Interface, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

ARTICLE INFO

Article history:
Received 12 July 2011
Received in revised form
13 April 2012
Accepted 7 May 2012
Available online 1 June 2012

Keywords:
Robot learning
Programming by demonstration
Real-time generalization of movements
Statistical methods
Gaussian process regression
Locally weighted regression
Dynamic movement primitives
Kinesthetic guiding
Example movements
Radial basis functions

ABSTRACT

Autonomous robots cannot be programmed in advance for all possible situations. Instead, they should be able to generalize the previously acquired knowledge to operate in new situations as they arise. A possible solution to the problem of generalization is to apply statistical methods that can generate useful robot responses in situations for which the robot has not been specifically instructed how to respond. In this paper we propose a methodology for the statistical generalization of the available sensorimotor knowledge in real-time. Example trajectories are generalized by applying Gaussian process regression, using the parameters describing a task as query points into the trajectory database. We show on real-world tasks that the proposed methodology can be integrated into a sensory feedback loop, where the generalization algorithm is applied in real-time to adapt robot motion to the perceived changes of the external world.

1. Introduction

In this paper we investigate the problem of real-time, goal-directed trajectory generation from a database of example movements. This problem has received a considerable amount of attention in recent years [1–5]. It has often been studied in the context of the programming by demonstration paradigm [6,7]. Our primary interest is in real-time synthesis of new trajectories using local, statistical methods. For the purpose of this paper, real-time is defined to be the frequency comparable to a typical camera stream, i.e. 30 Hz. In our experiments, the database of training movements has been acquired by sequentially guiding a robot through a set of example trajectories, but automatic approaches to segmentation from a long sequence of example movements are also possible [8].

It has been shown by Ude et al. [5] that it is possible to generalize the movements collected in an example database to new situations by utilizing the goal of an action as a query point into the database. In [5] movement generalization was implemented by employing a combination of locally weighted regression [9] and Gaussian process regression [10], where raw trajectory data was used as input for generalization. The proposed approach was applied to generalize various behaviors including reaching, throwing, and drumming. While this approach can take into account external perceptual feedback to generalize example movements to different situations, its computational cost is prohibitive for use in a real-time feedback loop. The goal of this paper is to provide an approach that is efficient enough to be applied in such a loop.

The approach described in [5] uses Dynamic Movement Primitives (DMPs) [11,12] as the basic representation for the encoding of robot movements. DMPs have many useful properties such as a built-in ability to react to perturbations without introducing discontinuities in the resulting robot motion. As an autonomous representation, they are not directly dependent on time, which makes it easy to stop the execution of movement without extensive bookkeeping of time evolution [13]. DMPs can also be extended to include capabilities such as obstacle avoidance [14] and avoidance of joint limits. All these adaptations can be done in real-time, which enables the robot to react to external sensory feedback. In this paper we expand on such built-in abilities by providing a methodology for real-time generation of DMPs using a trajectory database. In this way we provide means for on-the-fly, task-specific adaptation of motion.

One DMP can encode one specific robot trajectory. In case of point-to-point (discrete) movements, the trajectory of each robot degree of freedom y (given either in joint or in task space) is described by the following system of nonlinear differential equations

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x), \quad (1)$$

$$\tau \dot{y} = z, \quad (2)$$

$$\tau \dot{x} = -\alpha_x x, \quad (3)$$

* Corresponding author.
E-mail address: denis.forte@ijs.si (D. Forte).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.05.004

where x is the phase variable and z is an auxiliary variable. α_x, α_z, β_z and τ need to be specified in such a way that the system converges to the unique equilibrium point $(z, y, x) = (0, g, 0)$. The nonlinear term f contains free parameters that enable the robot to follow any smooth point-to-point trajectory from the initial position y_0 to the final configuration g

$$f(x) = \frac{\sum_{k=1}^{N} w_k \Psi_k(x)}{\sum_{k=1}^{N} \Psi_k(x)} x, \quad \Psi_k(x) = \exp\left(-h_k (x - c_k)^2\right). \tag{4}$$

Here c_k are the centers of radial basis functions distributed along the trajectory and $h_k > 0$. Weights w_k need to be estimated so that the DMP encodes the desired trajectory. For robots with many degrees of freedom, each degree of freedom is represented by its own equation system (1)–(2), but with a common phase (3). We used the algorithm described in [5] to determine the placement, width, and number of radial basis functions Ψ_k.

Recently, Gribovskaya et al. [2] proposed an alternative approach based on dynamic systems that can encode a complete class of movements. In their approach, a class of movements is represented by a more general nonlinear system of differential equations

$$\dot{\mathbf{y}} = \mathbf{h}(\mathbf{y}). \tag{5}$$

Note that the phase information is not available in this approach, which is problematic in the case of intersecting velocity fields, where the same robot configuration $\mathbf{y}$ is associated with more than one velocity $\dot{\mathbf{y}}$. Nevertheless, unlike DMPs, this representation is not limited to one specific movement but can represent more complete velocity fields that can encode a class of movements. The nonlinear function $\mathbf{h}$ can be estimated using Gaussian mixture regression [2], which results in a large-scale global optimization problem. Once this optimization problem is solved, the on-line robot control can be realized by integrating equation (5), which enables the robot to switch to a different movement (within the learned class of movements) in real-time. The stability analysis of the resulting system of differential equations is also given in [2].

Since DMPs have not been designed to represent whole classes of movements, a standard DMP cannot switch to a different type of movement in case of perturbations. It can only react to a certain perturbation by "pulling" the robot back to the desired trajectory in a generic fashion. While the recently proposed approach described in [5] does allow for the generation of DMPs that are adapted to a given configuration of the external world, this approach is computationally too expensive to be applied within the robot feedback loop. However, it has the advantage that the generalized DMP is computed by local regression methods, which makes it possible to apply local, linear optimization as compared to the global, nonlinear optimization approach described in [2]. Since the approach from [5] does not attempt to represent a whole class of movements within one differential equation, the selection of basis functions for regression is less critical than in [2]. In this paper we propose a new approach to movement generalization using Gaussian process regression, which not only preserves the advantages of the local approach, but also allows for real-time computation of DMPs, thus making it suitable for its application within a real-time sensory feedback loop.

Parametric hidden Markov models [3] are another movement representation that could be used to encode and parameterize a database of example movements forming a movement primitive. However, dynamic movement primitives are a more suitable movement representation than hidden Markov models when the execution on a real robot is paramount. This is due to the built-in ability of DMPs to generate smooth movements even in the case of perturbations. In addition, DMPs can be modulated to react in a generic way to different changes that may arise during the execution of the task.

2. Approximation of a class of movements with Gaussian process regression

Lets assume that we have a set of robot movements $\mathbf{M}_i$, $i = 1, \ldots, NumEx$, which all result in a successful execution of a given task in different situations. As an example we consider a set of reaching movements towards different targets in 3-D space. We denote the parameters characterizing the task by $\mathbf{q}_i \in \mathbb{R}^m$, $i = 1, \ldots, NumEx$, m being the dimensionality of these parameters, which we also call query points and $NumEx$ the number of example movements. Every movement $\mathbf{M}_i$ is encoded by a sequence of trajectory points $\{\mathbf{y}_{ij}, \dot{\mathbf{y}}_{ij}, \ddot{\mathbf{y}}_{ij} \in \mathbb{R}^{dof}\}$, measured at times t_{ij}, $j = 1, \ldots, n_i$, $t_{i1} = 0$. Here n_i denotes the number of samples on trajectory $\mathbf{M}_i$, while dof denotes the number of degrees of freedom encoded by the example trajectories. We have experimented both with end-effector trajectories (in this case $\mathbf{y}_{ij}$ are points in the Cartesian space) and with robot joint trajectories (in this case $\mathbf{y}_{ij}$ are the joint angles stemming from the active degrees of freedom). The problem is to compute a trajectory for any given query point $\mathbf{q}$. For example, in the case of reaching, a query point is given by the desired target position and we need to compute the associated reaching trajectory $\mathbf{M}$. Example movements $\mathbf{M}_i$ can be acquired either by kinesthetic guiding [15] or by imitation [16].

To become able to accomplish a task in any situation, the robot needs to learn a function that maps the parameters describing the task $\mathbf{q}$ into the parameters describing the desired trajectory $\mathbf{M}$, i.e.

$$\mathbf{G} : \mathbf{q} \longmapsto \mathbf{M}. \tag{6}$$

In general, $\mathbf{G}$ is not a function. For example, in the case of reaching movements, there are many different ways to reach towards a desired destination. However, we can impose an additional constraint that synthetic reaching trajectories should be similar to the example reaching trajectories. The closer the desired query point $\mathbf{q}$ is to the example query point $\mathbf{q}_j$, the more similar the generated trajectory $\mathbf{M}$ should be to the trajectory $\mathbf{M}_j$ associated with query point $\mathbf{q}_j$. With this additional constraint, $\mathbf{G}(\mathbf{q}; \{\mathbf{M}_1, \ldots, \mathbf{M}_{NumEx}\})$ becomes a function that can be learned.

Our approach is based on the assumption that humans are unlikely to use a totally different movement strategy if the task parameters change only slightly and there are no qualitative changes in the task. Under such circumstances, human teachers normally generate sets of training movements that smoothly transition from one to another. It is therefore reasonable to assume that transitions between movements that solve the task in slightly different situations are smooth as long as there are no qualitative changes in the task. For our reaching example this means that if the training trajectories are given in the joint space and the robot is redundant, then the teacher must select the same inverse kinematics solution in all training movements. It is not possible to mix qualitatively different joint trajectories because statistical generalization cannot be aware of the kinematics properties of the robot.

2.1. Converting the example trajectories into dynamic movement primitives

To reduce the amount of data that we need to process for action generalization, we first encode each of the example movements $\mathbf{M}_i$ as a dynamic movement primitive (DMP). Any of the standard methods proposed in the literature can be used for this purpose. Let's denote

$$f_{ijl}^{targ} = \tau_i^2 \ddot{y}_{ijl} - \alpha_z(\beta_z(g_{il} - y_{ijl}) - \tau_i \dot{y}_{ijl}), \tag{7}$$

where $i = 1, \ldots, NumEx$, $j = 1, \ldots, n_i$, $l = 1, \ldots, dof$. The above equation can be derived by rewriting the system of two first-order differential equations (1)–(2) as one second-order

differential equation (by writing $z = \tau\dot{y}, \dot{z} = \tau\ddot{y}$ in (1)). The parameters w_{ikl}, $k = 1, \ldots, N$ (N is the number of DMP kernel functions, see (4)) can be estimated by solving the following linear regression problems

$$\mathbf{X}_i \mathbf{w}_{il} = \mathbf{f}_{il}^{targ}, \tag{8}$$

where

$$\mathbf{X}_i = \begin{bmatrix} \frac{\Psi_1(x_{i1})}{\sum_{i=1}^{N} \Psi_i(x_{i1})} x_{i1} & \cdots & \frac{\Psi_N(x_{i1})}{\sum_{k=1}^{N} \Psi_i(x_{i1})} x_{i1} \\ \cdots & \cdots & \cdots \\ \frac{\Psi_1(x_{iT})}{\sum_{k=1}^{N} \Psi_k(x_{iT})} x_{iT} & \cdots & \frac{\Psi_N(x_{iT})}{\sum_{k=1}^{N} \Psi_k(x_{iT})} x_{iT} \end{bmatrix}, \tag{9}$$

and $\mathbf{w}_{il} = [w_{i1l}, \ldots, w_{iNl}]^T$, $\mathbf{f}_{il} = [f_{i1l}^{targ}, \ldots, f_{iTl}^{targ}]^T$. The phase parameters $x_{ij} = x(t_{ij})$ are calculated by integrating equation (3) with the boundary condition $x_{i1} = x(0) = 1$.

The above process enables us to convert the initial raw trajectory data $\mathbf{M}_i$ into DMPs, i.e. $\mathbf{M}_i \longmapsto (\mathbf{w}_i, \mathbf{g}_i, \tau_i)$, where $\mathbf{w}_i \in \mathbb{R}^{N \times dof}$ are the weights of DMPs for all degrees of freedom, $\mathbf{g}_i \in \mathbb{R}^{dof}$ are the final configurations on the example trajectories, i.e. $\mathbf{g}_i = \mathbf{y}_{in_i}$, and $\tau_i \in \mathbb{R}$ are the time durations of example trajectories, i.e. $\tau_i = t_{in_i}$.

2.2. Trajectory generalization using Gaussian process regression

The conversion of raw example trajectories into DMPs results in a significant data reduction. For example, a four second trajectory sampled at 500 Hz contains 2000 data points, which can typically be reduced to a DMP defined by a few tens of radial basis functions. In this section we propose to synthesize new movements directly from the estimated DMP parameters. In this case function (6) becomes

$$\mathbf{G}\left(\{\mathbf{w}_i, \mathbf{g}_i, \tau_i; \mathbf{q}_i\}_{i=1}^{NumEx}\right) : \mathbf{q} \longmapsto (\mathbf{w}, \mathbf{g}, \tau)\,. \tag{10}$$

Gaussian Process Regression (GPR) can be applied to estimate function (10). Gaussian processes are based on Bayesian probability modeling [10]. The resulting models have an interesting and useful feature that, besides output values, they also predict confidence in these values. GPR exhibits good generalization performance and the predictive distribution can be used to measure the uncertainty of the estimated function. It has been demonstrated that this technique outperforms other regression methods on problems such as estimating inverse dynamics of a seven degrees of freedom robot arm [17].

Technically, a Gaussian process is defined as

$$g(\mathbf{q}) \sim \mathcal{GP}\left(m(\mathbf{q}), k(\mathbf{q}, \mathbf{q}')\right), \tag{11}$$

where $m(\mathbf{q}) = \mathbb{E}[g(\mathbf{q})]$ is the mean function and $k(\mathbf{q}, \mathbf{q}') = \mathbb{E}[(g(\mathbf{q}) - m(\mathbf{q}))(g(\mathbf{q}') - m(\mathbf{q}'))]$ the covariance function of the process. Let's assume that we have – as when estimating function (10) – a set of noisy observations $\{(\mathbf{q}_i, y_i) | i = 1, \ldots, NumEx\}$, $y_i = g(\mathbf{q}_i) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. Subtracting the mean from the training data, we can further assume that $m(\mathbf{q}) = 0$. Given a set of query points $g(\mathbf{q}^*)$, the joint distribution of all outputs is estimated by

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I} & \mathbf{K}(\mathbf{Q}, \mathbf{Q}^*) \\ \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}) & \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}^*) \end{bmatrix}\right), \tag{12}$$

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 Eq. (11). It can be shown [10] that the expected value $\bar{\mathbf{y}}^*$ associated with the new query points $\mathbf{q}^*$ is given by

$$\bar{\mathbf{y}}^* = \mathbb{E}[\mathbf{y}^* | \mathbf{Q}, \mathbf{y}, \mathbf{Q}^*] = \mathbf{K}(\mathbf{Q}^*, \mathbf{Q})[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}, \tag{13}$$

with the following estimate for the covariance of the prediction

$$\mathrm{cov}(\mathbf{y}^*) = \mathbf{K}(\mathbf{Q}^*, \mathbf{Q}^*) - \mathbf{K}(\mathbf{Q}^*, \mathbf{Q})[\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}(\mathbf{Q}, \mathbf{Q}^*).$$

One commonly used covariance function is

$$k(\mathbf{q}, \mathbf{q}') = \sigma_f^2 \sum_{i=1}^{m} \exp\left(-\frac{1}{2} \frac{(q_i - q_i')^2}{l_i^2}\right), \tag{14}$$

which results in a Bayesian regression model with an infinite number of basis functions. m denotes the dimension of the query point space. See [10] for more details.

With GPR new estimates are calculated using Eq. (13). The most computationally expensive part 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. The dimension of this matrix is equal to the number of data points. In our case, this is equal to the number of example movements *NumEx*, which is typically not too large (at most a few hundred). We thus only need to invert a matrix of dimension *NumEx* × *NumEx*.

Note that by writing

$$\mathbf{z} = [\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}. \tag{15}$$

Eq. (13) and the estimated parameter $\bar{y}^*$ associated with the query $\mathbf{Q}^* = \mathbf{q}^*$ can be written as

$$\bar{y}^* = \sum_{i=1}^{NumEx} k(\mathbf{q}^*, \mathbf{q}_i) z_i, \tag{16}$$

where in our experiments $\bar{y}^*$ stands for $\bar{\tau}^*$, $\bar{g}_l^*$, and $\bar{w}_{kl}^*$. Thus the training data are weighted based on the distance between the training query points and the current query point. This means that nearby training points influence the result more. In this sense, GPR can be viewed as a local regression method.

To generate a new movement, the robot is given a desired query point $\mathbf{q}^*$, e.g. the desired reaching location. For each of the parameters defining a generalized DMP (τ, g_l, and w_{kl}), which encodes a suitable motion trajectory for this task situation, we need to calculate (16) on-line, whereas (15) can be stored in memory. Note also that matrix $\mathbf{K}(\mathbf{Q}, \mathbf{Q}) + \sigma_n^2 \mathbf{I}$ depends only on query points and not on the data points. Thus this matrix is the same for all parameters defining a DMP and thus needs to be inverted only once.

2.3. Comparison with previous local approaches

In our experiments we compare the performance of the proposed approach and the performance of a method that uses complete trajectories without an intermediate trajectory conversion step (as proposed in [5]) for the purpose of task-specific generalization of dynamic movement primitives. In this section we examine the advantages and disadvantages of both approaches and their suitability for on-line generalization.

In contrast to our new approach, which converts the training data into DMPs, the approach proposed in [5] keeps complete trajectories in memory and generalizes to new DMPs without the intermediate trajectory conversion step. In this case locally weighted regression (LWR) instead of Gaussian process regression has to be used in some calculations that are needed for trajectory generalization. LWR is a memory-oriented, non-parametric method for statistical approximation. The basic idea is to compute local models using data from a small neighborhood of the desired query point.

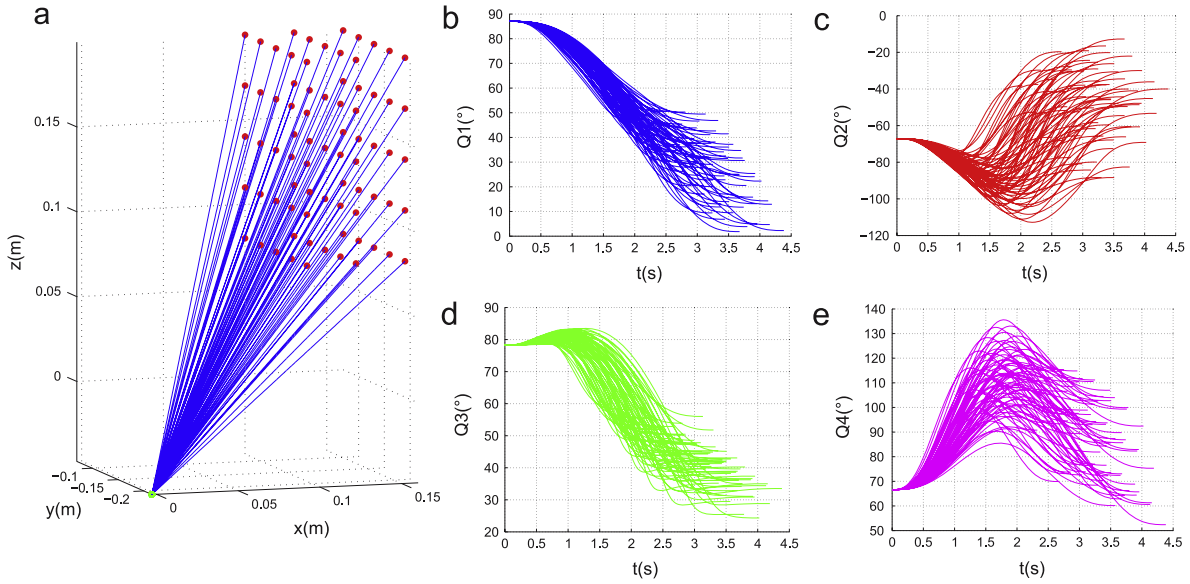


Fig. 1. 75 minimum jerk trajectories were used as training data to test the generalization abilities of our approach. The 3-D graph (a) shows the generated Cartesian space minimum jerk trajectories and 2-D graphs show the associated joint trajectories. (b) shows shoulder flexion–extension, (c) shoulder abduction–adduction, (d) upper arm rotation and (e) elbow flexion–extension of the right arm of the HOAP-3 robot.

Since raw trajectories are used for estimation, the resulting systems of linear equations are much too large to be resolved on-line. Unlike this approach, which defers most of the calculations to the moment when a query needs to be answered, our new, GPR-based method performs most of the calculations off-line once all of the training data have been acquired. The most expensive off-line calculations are needed for the calculation of (15) and for the estimation of hyper-parameters l_i, σ_f and σ_n as defined in (13) and (14). After the end of learning the training data can be discarded and only simple calculations shown in (16) are needed to answer a new query or – in other words – generalize to new situations.

Here we note that in the case of LWR we need to specify one additional parameter, i.e. influence radius, which determines how many nearby trajectories will be taken into account for generalization by LWR. Some approaches for the selection of the optimal radius can be found in [5]. Our new, GPR-based approach does not require such a parameter. In the following we call the approach proposed in this paper *MPG* (Movement Primitives Generalization) and the approach from [5] *RTG* (Raw Trajectories Generalization).

3. Experimental results

3.1. Simulation results

We first show how good the proposed method is at generalizing motor knowledge, which was created in simulation. For this test we generated 75 minimum jerk trajectories in Cartesian space with zero velocity and acceleration at the beginning and at the end of each movement. It has been shown that minimum jerk trajectories are similar to human arm point-to-point reaching movements [18]. The simulated Cartesian space trajectories were converted into joint space trajectories of the right arm of humanoid robot HOAP-3 using inverse kinematics that selected a specific kinematic solution. This resulted in 75 four-dimensional joint space trajectories, which were used as training data. The Cartesian end-positions of trajectories were employed as query points. The query points were uniformly distributed 3 cm apart within a cuboid with dimensions 6 × 12 × 12 cm (see Fig. 1). In our tests we compared how close the generalized trajectories are to the ideal minimum jerk trajectories and how the proposed method compares to the method developed in [5], which uses complete trajectories as input data without the intermediate trajectory conversion step.

We tested the performance of both approaches by calculating the generalized trajectories at query points different from the training queries. We compared the generalized joint space trajectories with the ideal minimum jerk trajectories in the Cartesian space. In summary, our experiments show that MPG calculates the generalized trajectory much faster but with a slightly larger deviation than RTG. In the following, we analyze the error in the calculation of DMP goals $\mathbf{g}$ and the errors over the complete course of the trajectories estimated by the two different approaches.

As evident in (10), at each query point $\mathbf{q}$ we need to calculate the DMP parameters $\mathbf{w}, \mathbf{g}$, and τ. In our simulation example, query points are the end-points on the trajectories given in the Cartesian space and $\mathbf{g}$ are the associated joint angles at the end of the corresponding joint space trajectories. Thus in this case GPR attempts to estimate one particular solution of the inverse kinematics of the arm of HOAP-3. For testing, query points in Cartesian space given to GPR were uniformly distributed within the training cuboid with a distance of 1 cm between them. The error was measured by calculating the generalized goal $\mathbf{g}$ $\left(\mathbf{g}_{joint,gen} = \mathbf{G}\left(\{\mathbf{w}_i, \mathbf{g}_i, \tau_i; \mathbf{q}_i\}_{i=1}^{NumEx}; \mathbf{q}_{xyz}\right)\right)$, transforming the generalized goal from the joint space back to the Cartesian space using known forward kinematics of the arm ($\mathbf{q}_{xyz,gen} = FK(\mathbf{g}_{joint,gen})$), and calculating the distance between this transformed position and the original query point $e = \|\mathbf{q}_{xyz,gen} - \mathbf{q}_{xyz}\|$. The average difference between these points was 0.8 mm (see also Fig. 2), which is significantly smaller than the distance between the training data (3 cm). The analysis of Figs. 2 and 3(a), which depicts the training points (in blue), shows that the minimal errors appear exactly in the planes that contain the training queries. Fig. 2 also shows that the error increases if the query points are between the training points. Finally, extending the generalization algorithm beyond the training area results in a rapidly increasing error due to extrapolation.

In simulation we also tested the accuracy of the complete generalized trajectories as estimated by MPG and RTG. We compared the results of both approaches with the ideal minimum jerk trajectories in joint space over the entire trajectories, which

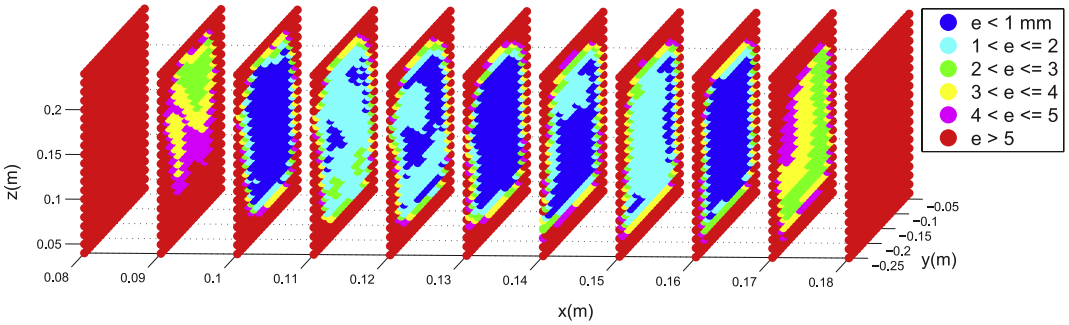


Fig. 2. Error in the estimation of the DMP goal parameter **g** (see also the text). Dark blue areas in the graph represent estimation error of less than 1 mm, light blue areas error between 1 and 2 mm, green between 2 and 3 mm, yellow between 3 and 4 mm, rose between 4 and 5 mm, and red above 5 mm. The average error inside the training space ($x \in [0.1, 0.16]$, $y \in [-0.19, -0.07]$, $z \in [0.08, 0.2]$) is 0.8 mm. The error increases rapidly outside of the training space.

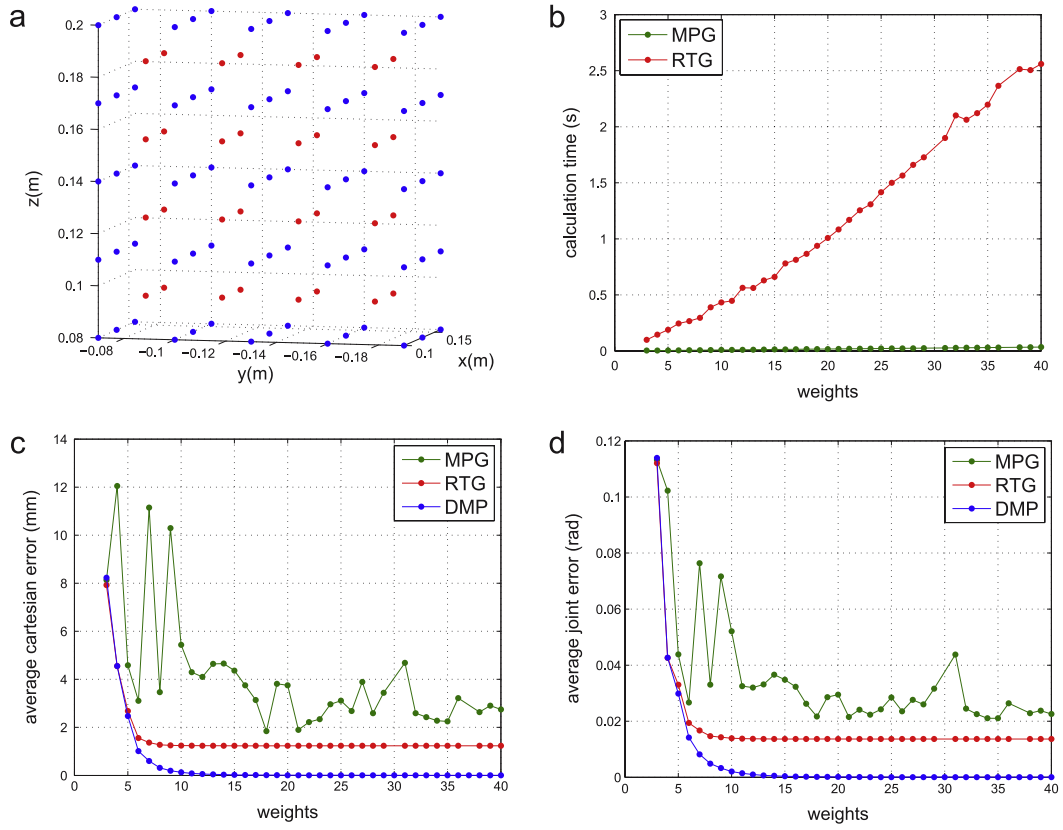


Fig. 3. Results of MPG and RTG were compared with the ideal minimum jerk trajectories (in joint space) ending in query points (red dots) situated between the training points (blue dots) as shown in graph (a). The blue line in graph (c) shows the error in Cartesian space of the reconstructed minimum jerk trajectories in dependence on the number of weights (here the error is due to inaccurate approximation by DMPs), the red line shows an error of RTG, and the green line shows an error of MPG. Graph (d) depicts the error in joint space and graph (b) shows calculation times that are needed to perform RTG and MPG. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

end in query points (red dots in Fig. 3(a)) situated between the training points (blue dots in Fig. 3(a)). We also tested how the approximation by a different number of weighted radial basis functions affects the average trajectory error:

$$error_k = \frac{1}{T_{end,k}} \sum_{i=1}^{T_{end,k}} \|\mathbf{y}_{i,k,gen} - \mathbf{y}_{i,k,ideal}\|,$$

$$k = 1, \ldots, NumEx, \tag{17}$$

$$error_W = \frac{1}{NumEx} \sum_{k=1}^{NumEx} error_k, \tag{18}$$

where $\mathbf{y}_{i,k}$ are the points on the trajectory k, given either in Cartesian or in joint space.

Results shown in Fig. 3(c) and (d) demonstrate that RTG reaches the minimum error sooner and is more stable than MPG, but MPG also reaches a comparable error minimum when a few more radial basis functions are used to encode the DMPs. Fig. 3(b) shows a big difference in computation times for MPG and RTG as the number of DMP basis functions increases. MPG generalizes much faster and is therefore more suitable for on-line calculation. These results also show that 18 radial basis functions are enough to approximate the simulated reaching trajectories. With more than 18 weights the average error does not change significantly regardless of the selected method. Fig. 4 shows how well some of the generalized trajectories (encoded with 18 radial basis functions) fit the ideal minimum jerk trajectories.

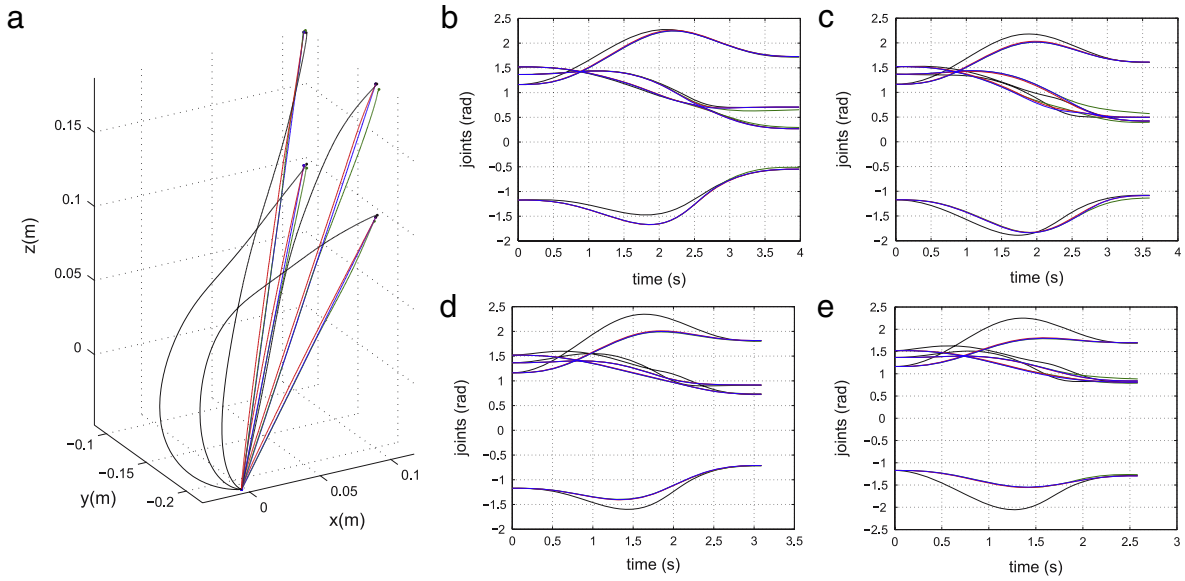


Fig. 4. The 3-D graph (a) shows a few calculated minimum jerk trajectories compared with generalized trajectories using MPG and RTG as well as the movement generated by one training DMP where only the goal of the movement was changed (weights and τ were not generalized). Blue curves are the real minimum jerk trajectories, green curves are the generalized trajectories estimated by MPG, red curves are the generalized trajectories generated by RTG, and black curves are the trajectories generated by a DMP with constant weights but different goals. 2-D graphs (b), (c), (d) and (e) show all these trajectories in joint space (as a function of time). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In cases when only one DMP is available, reaching movements toward arbitrary points in space can still be generated by modulating the goal parameter $\mathbf{g}$ of the available DMP. However, the modulated movements do not retain the shape of the original movement as encoded by the DMP. This is a problem if the course of movement is important. By utilizing multiple training examples representing a class of reaching movements and statistical generalization, we can generate DMPs that lead the robot along trajectories of a specific shape. This is especially important if movements that avoid a certain obstacle need to be generated. In such cases just the modulation of the goal position without weights generalization almost always results in a collision with the obstacle.

The number of radial basis functions needed to approximate the trajectories depends on the type of movement. With MPG we need to use the same number of basis function to estimate all training trajectories, otherwise it is not possible to apply Gaussian process regression. The longer and more complex the trajectories are, the more radial basis functions are needed to approximate the movements. The automatic selection of the number of basis functions is discussed in [19].

To test the MPG and RTG in real experiments, where the correct trajectories are not known, we applied the leave-one-out cross validation (L1OCV) to determine the number of necessary basis functions. In L1OCV, each of the demonstrated trajectories is taken out from the training data and re-estimated by generalization from the remaining trajectories. The generalized and the skipped trajectory are then compared to determine an average error over the entire trajectory. The L1OCV score is given by an average error over all trajectories in the training data. To make comparison with real experiments easier, we also tested the leave-one-out cross validation with the simulated trajectories (see Fig. 5).

Results in Fig. 5 show that the average Cartesian and joint space L1OCV scores follow a similar pattern as in Fig. 3. The differences are due to the different distribution of training and test query points (here the test query point is always as far away as possible from the training points, whereas in the previous simulation experiment we tested the full distribution of query points).

3.2. Reaching and grasping with a humanoid robot HOAP-3

The aim of our next experiment was to show that our method can be used to teach the humanoid robot HOAP-3 how to reach for an object and grasp it using data coming from its own visual system. The training was done by collecting a number of example reaching movements using kinesthetic guiding along the desired reaching trajectories. We held the robot's arm near the right elbow joint and manually moved it from the initial position to the desired end-points in front of the robot (Fig. 6). All four joints of the right arm were collected along the demonstrated trajectories. Altogether we collected 140 training movements to 3-D locations that the robot can see with its cameras. Endpoints were about 5 cm apart from each other distributed in the workspace of the robot's right arm (Fig. 7), which is approximately $30 \times 30 \times 10$ cm.

Besides the trajectories we also measured the final reaching positions as estimated by the active stereo vision of the robot. The positions estimated by vision were used as query points. Despite accurate calibration of the active camera system of a humanoid robot, some noise and errors are to be expected [20]. By comparing the results of vision-based 3-D position estimation with the results of the robot's forward kinematics, we estimated the systematic vision error to be 1.8 cm on average. However, the systematic vision error can be in part learned by Gaussian process regression. To collect the vision data, we put a small, colored spherical object into the robot hand and estimated its position at the final configuration on the trajectory (see the right graph in Fig. 7). Since stereo vision is also used to estimate the object position when generating a new movement, the vision errors that arise in training and the errors in query points used for generalization cancel each other out.

By changing the signs of all four joints of the right arm, we were able to transfer the training movements from the right to the left robot arm (Fig. 7, left). Such simplification was possible due to the design of the arm of the humanoid robot HOAP-3. Note that it is still necessary to estimate the end positions on the trajectories by vision because the error in the estimation of the object position depends on the configuration of the robot.

After collecting the training trajectories, we calculated the associated dynamic movement primitives (DMPs). The estimated

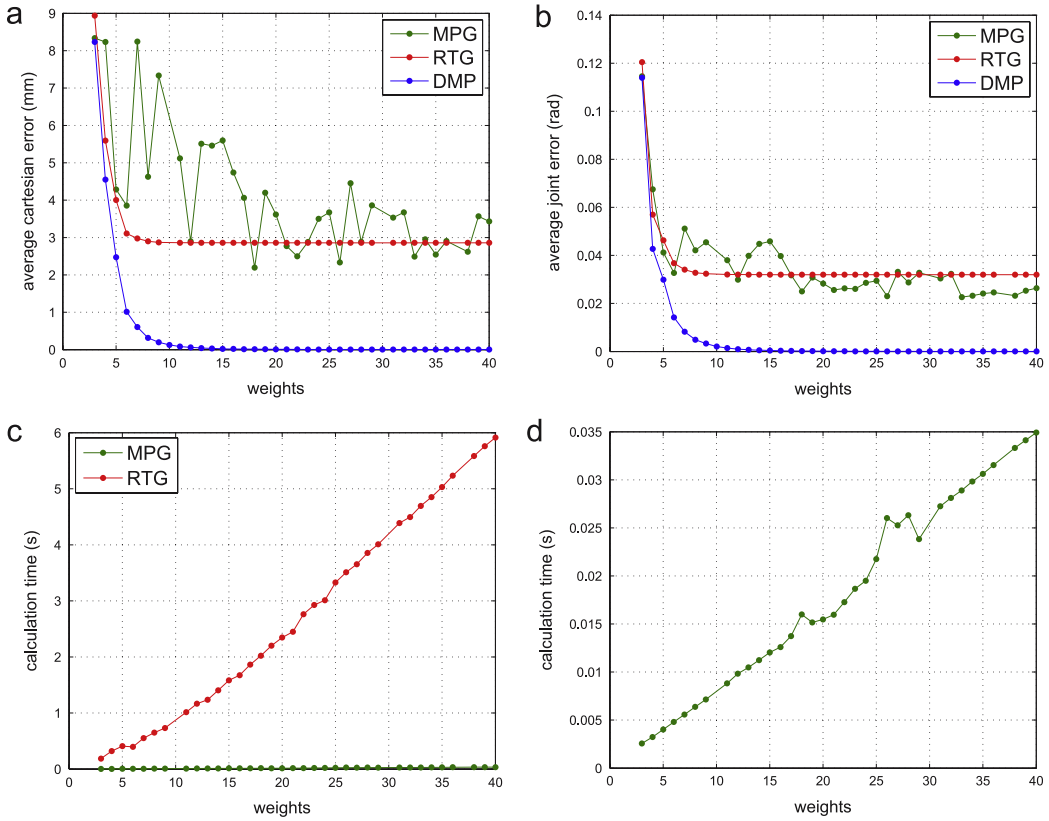


Fig. 5. The evaluation of MPG and RTG using L1OCV method. The blue line in graph (a) shows the Cartesian space error of DMPs encoding the simulated minimum jerk trajectories in dependence on the number of weights, the green line shows the error of generalization by MPG, and the red line depicts the error of generalization by RTG. Graph (b) shows the same errors in joint space, graph (c) shows the computational times needed for generalization by MPG and RTG. Graph (d) takes a closer look at calculation times of MPG. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

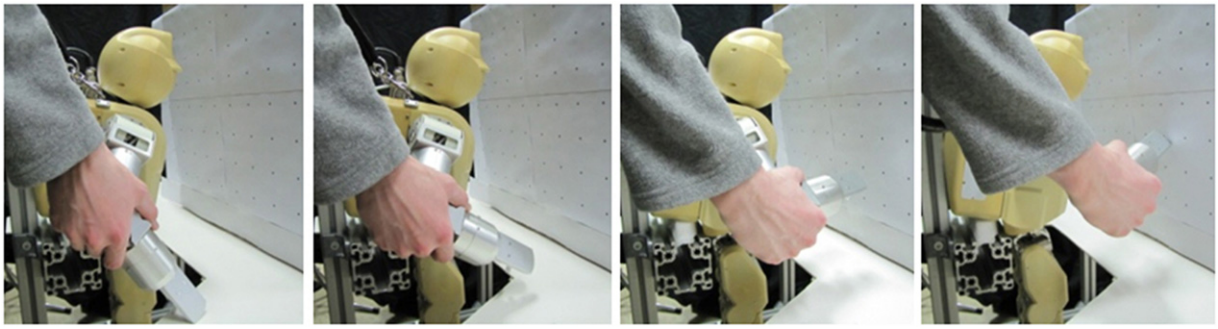


Fig. 6. Image sequence showing the acquisition of one reaching movement on humanoid robot HOAP-3 using kinesthetic guiding. We moved the arm towards the plate in front of the robot with points plotted at a distance of five centimeters between them. We gradually moved the plate away from the robot and demonstrated a series of movements at four different distances from the robot.

DMP parameters (weights associated with radial basis functions, time durations, and end-points of all reaching trajectories in joint coordinates) were used as input for learning by Gaussian process regression. To define how many radial basis functions were needed to properly approximate the acquired reaching movements, we used leave-one-out cross validation as explained above. Results can be seen in Fig. 8. They are similar to our simulation results. Again, the calculation time increases with more radial basis functions defining a DMP. Based on these results we used 27 radial basis functions to define DMPs because with a larger number of weights the L1OCW score does not change significantly. A comparison between the demonstrated trajectories, which were omitted from the training data, and trajectories generalized by MPG and RTG can be seen in Fig. 9.

Given a new desired reaching position in Cartesian coordinates (as estimated by the cameras), i.e. a new query point, the robot calculates the DMP parameters using Gaussian process regression as described in Section 2. The generated DMP is then integrated with Euler's method and the integration results are used to control the robot arm. If the object appears anywhere in the robot's workspace, HOAP-3 reaches towards the object with a movement similar to the demonstrated movements (Fig. 10).

When the robot hand reaches the final position, it grasps the object and moves back to the initial position. If the object position changes while the arm is moving, the robot uses its vision to estimate the new object position and calculates the new goal parameters using Gaussian process regression. The form of the movement trajectory remains similar to the training trajectories until the end of the estimated duration time; afterwards it takes the form dictated by the critically damped system (1) (the nonlinear part is close to zero once the time exceeds the estimated duration). If the robot has not reached the object within the expected time,

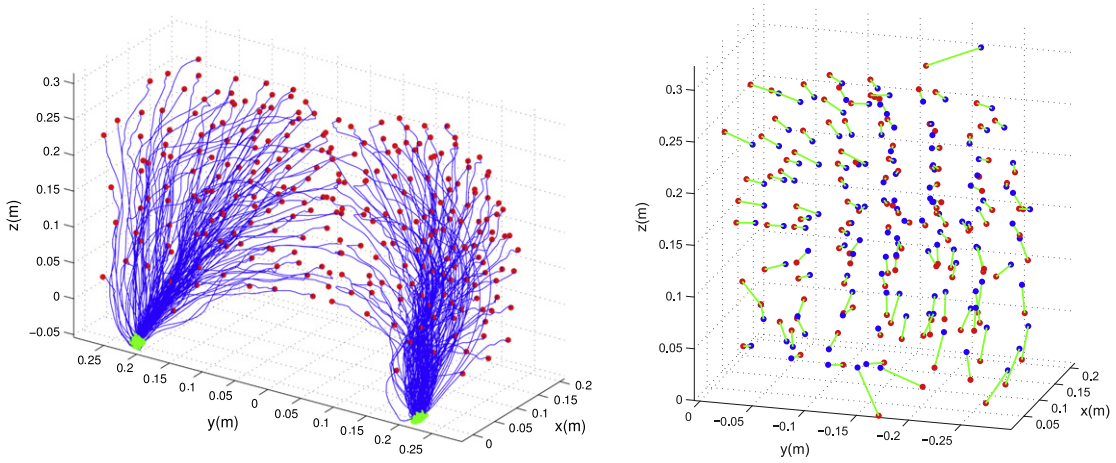


Fig. 7. The left graph shows example reaching trajectories. The final positions on the trajectories are shown as red dots and the starting positions as green dots. The trajectories for the left arm of the HOAP-3 robot were created by changing the signs of all four joints of the right arm. In the right graph red dots depict the original end-points of the right arm of all demonstrated trajectories as calculated by the robot's forward kinematics, while the blue dots are the same end-points as estimated by stereo vision. The green lines illustrate the shift of end-points, which is 1.8 cm on average. This represents a systematic error of stereo vision. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

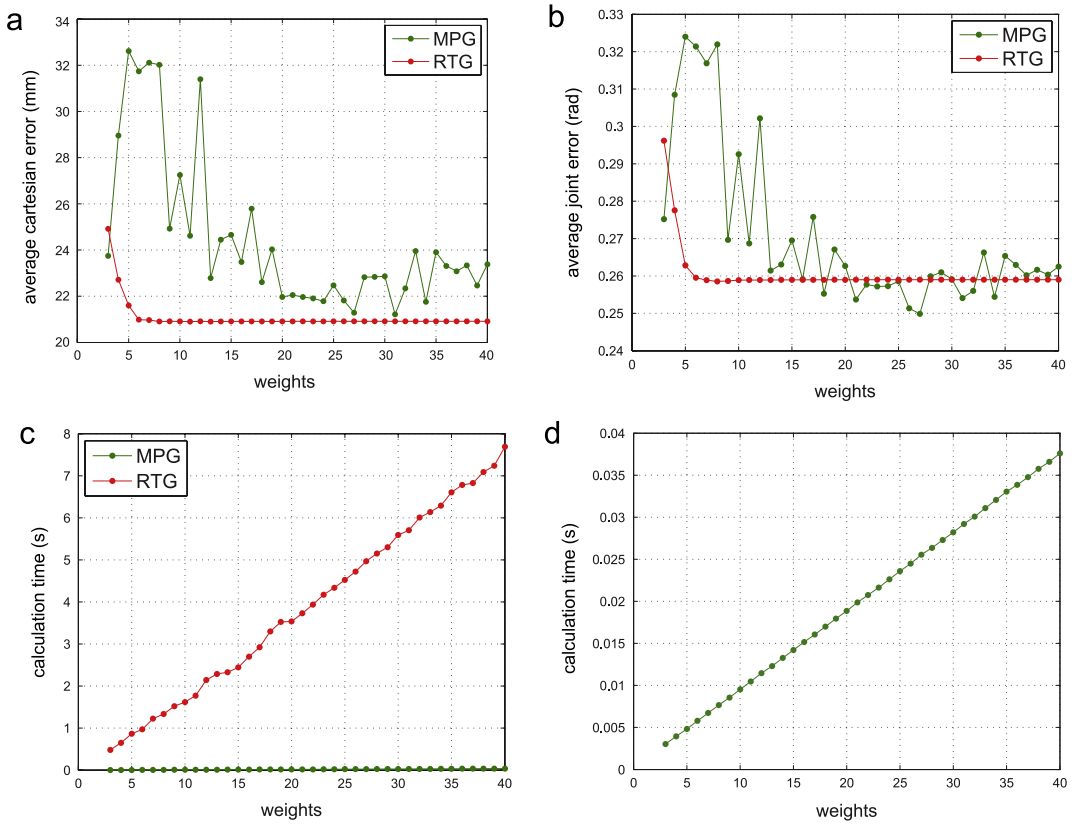


Fig. 8. The results of MPG and RTG were compared with the example trajectories by leave-one-out cross validation method. In graph (a), the red line shows L1OCW score obtained by RTG and the green line represents an L1OCW score obtained by MPG, all in Cartesian space. Graph (b) represents the L1OCW score in joint space and graph (c) shows calculation times that are needed for MPG and RTG, all in dependence on the number of weights. Graph (d) takes a closer look at calculation times of MPG. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

we continue to estimate the goal parameters **g** by vision and GPR, which ensures that the robot eventually reaches the object.

Figs. 11 and 12 show two reaching examples where the robot motion is continuously adapted to the current situation. In this experiment we show the integration between the proposed approach and walking to ensure that the robot can grasp an object. A similar experiment on a humanoid robot but based on more classic robotics approaches has been reported in [21]. If the object falls outside of the robot's workspace, HOAP-3 estimates the distance between the object and the robot's base coordinate system with its cameras and starts walking to come closer to the object. A predefined walking pattern was used for this purpose. While walking, the robot tracks the object and keeps estimating the distance to the object. This information is used to estimate how many intermediate steps are needed to reach the goal. The

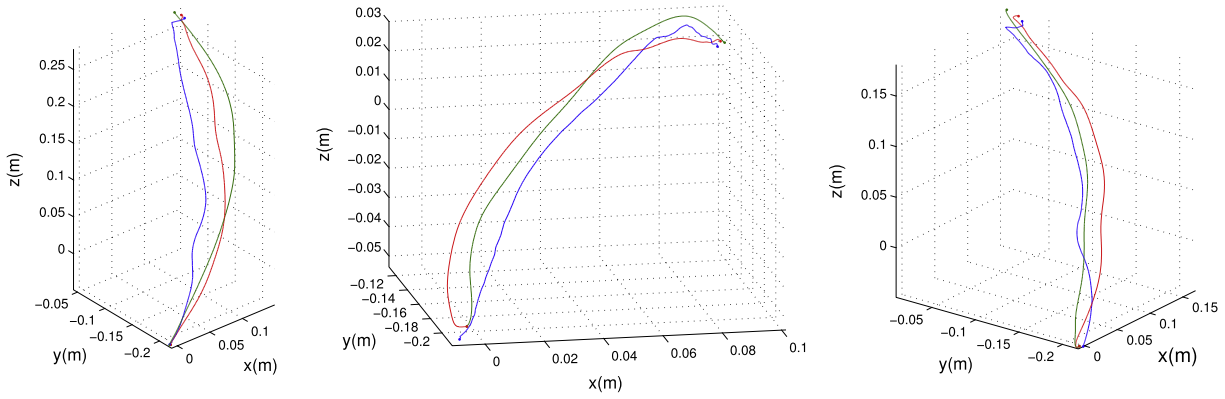


Fig. 9. A few examples comparing the demonstrated trajectories (blue curves), which were omitted from the training data, and trajectories calculated by MPG (green curves) and by RTG (red curves). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

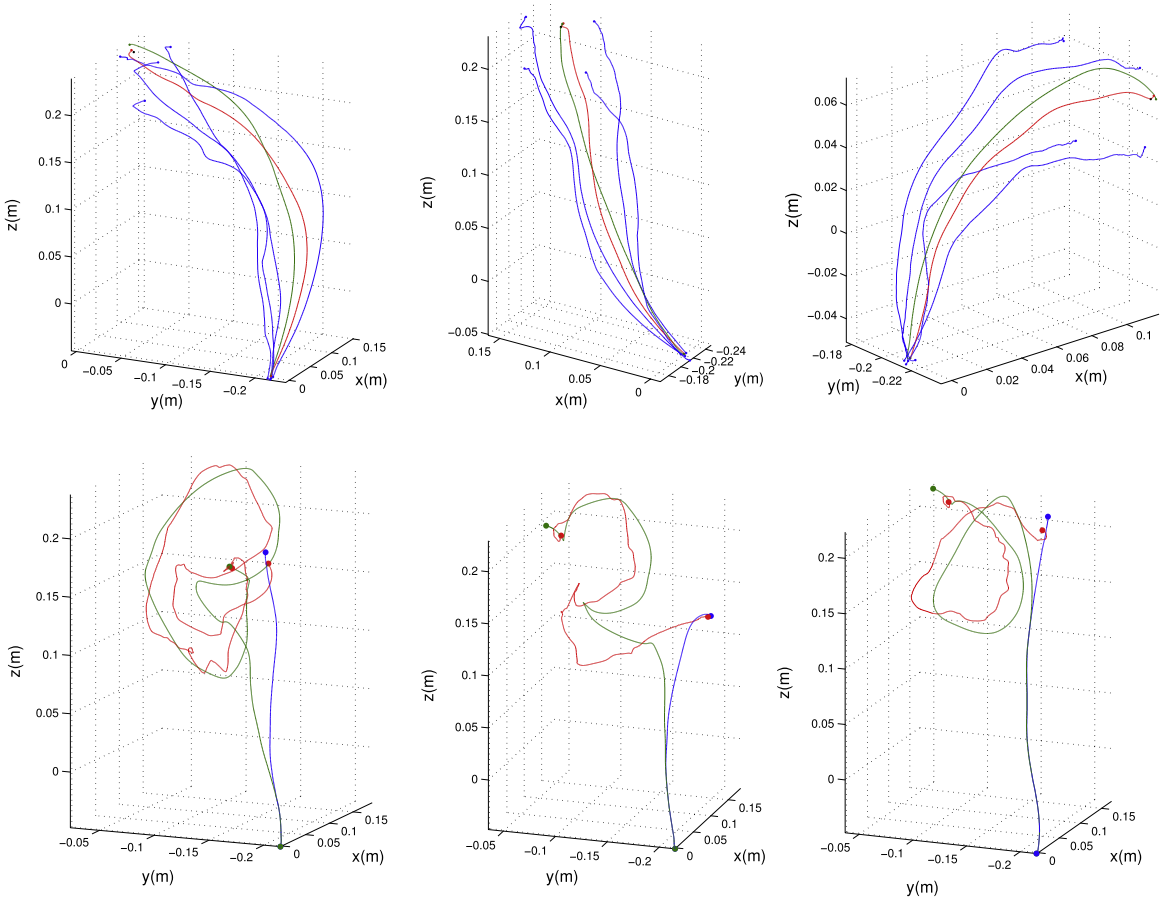


Fig. 10. *Top row*: the blue trajectories are nearby demonstrated trajectories, the green curves are the generalized trajectories estimated by MPG, and the red curves are generated by RTG. Notice the shape similarity between these trajectories. *Bottom row*: the blue trajectories show the generalized right hand movement if the object does not move. The green trajectories depict the actual right hand movement, which changed because the attractor point **g** was continuously adapted using the results of vision and Gaussian process regression. The red trajectories show the movement of the object as estimated by stereo vision. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

robot also adapts its orientation to ensure that it walks towards the object and finally grasps it.

As mentioned previously, the systematic error of stereo vision was partly learned by GPR and was consecutively reduced, but the error due to imperfect generalization by GPR remains. To test the generalization accuracy of GPR in a real experiment, the error of the method was estimated in the same way as in the simulation example. The test points, where the regression error was measured, were distributed on a regular grid with a distance of 1 cm between the end points inside the right arm workspace. Fig. 13 shows the difference between test points and 3-D position of those points as estimated by Gaussian process regression and robot forward kinematics. Here the error of GPR is 7.3 mm on average, which is low enough for the given task of grasping. The error in the case of real data is larger than in simulation, but this can be expected because the distribution of the simulated data was more compact, more regular and there was no noise. The error due to inaccuracy of GPR can be reduced by providing more training

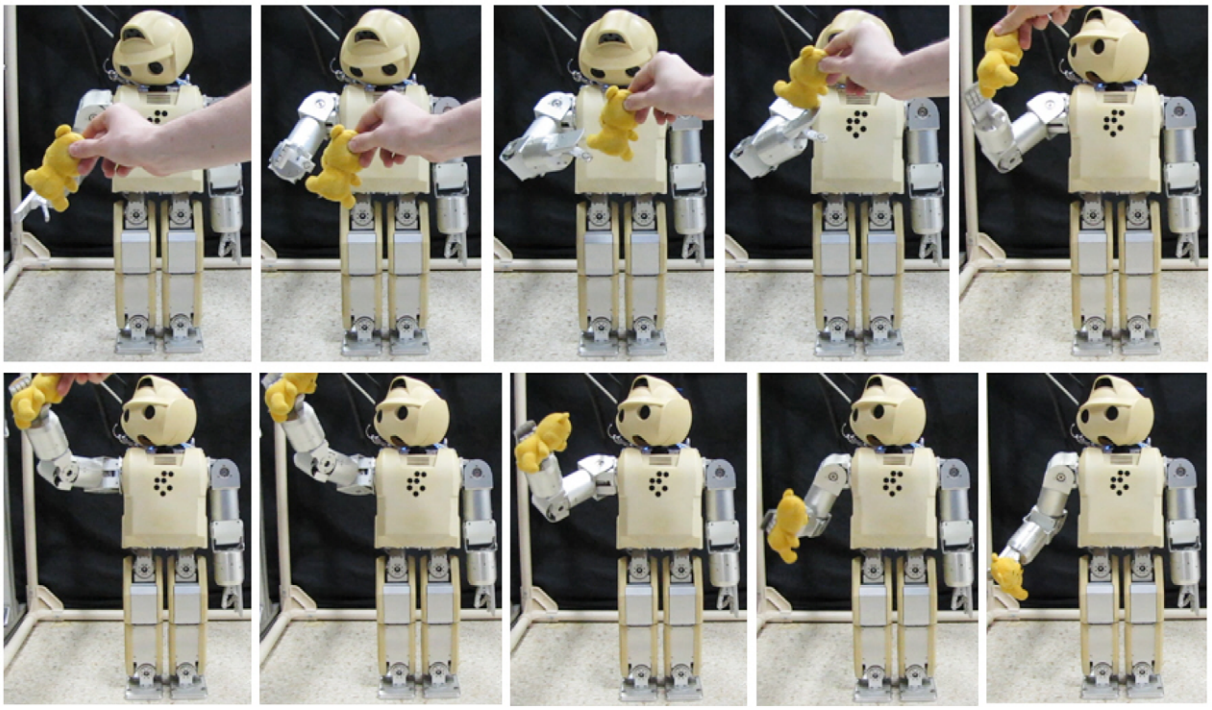


Fig. 11. HOAP-3 detects the object and keeps tracking it. Once the object stops moving, the robot can grasp it and move the arm back to its starting position.

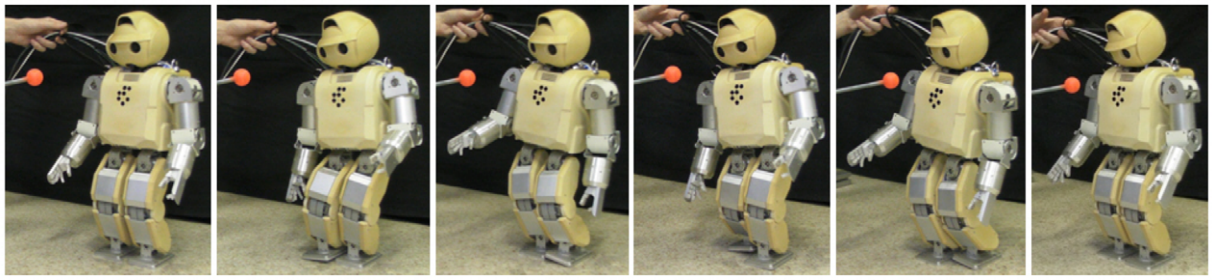


Fig. 12. If the object is outside of the robot's workspace, HOAP-3 uses vision to estimate the distance between the object and its base coordinate system and starts walking to come closer to the object. A predefined walking pattern was used for this task.

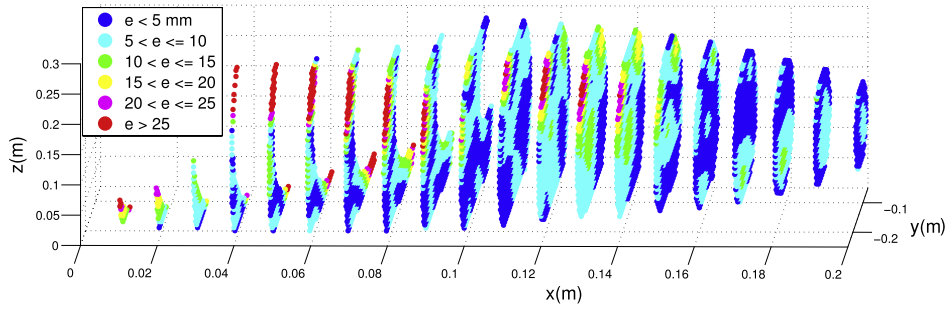


Fig. 13. Dark blue fields in the graph represent GPR error of less than 5 mm, light blue between 5 and 10 mm, green between 10 and 15 mm, yellow between 15 and 20 mm, rose between 20 and 25 mm and above 25 mm error are the red fields. The average error of this real-task endpoints is 7.3 mm.

data. However, a larger number of training examples requires more time and effort to acquire, so there is a trade-off between accuracy and training time.

3.3. Switching between two different movement primitives

Our final experiment was performed with the 7 DOF KUKA Light-Weight Robot arm. The main goal of this experiment was to demonstrate on-line generalization of trajectories, which can be accomplished only with our new approach (MPG) because RTG is too slow. The task was to reach towards an object and grasp it. The object can be grasped either from its right or from its left side. Thus the robot needs to learn how to grasp the object from both sides. We demonstrated 144 reaching movements (72 from the right side and 72 from the left side of the object), which were all acquired by kinesthetic guiding of the arm (see Fig. 14). For this task, the DMP-encoded training data was given in joint space.

Unlike in the experiments with HOAP-3, initial points on the

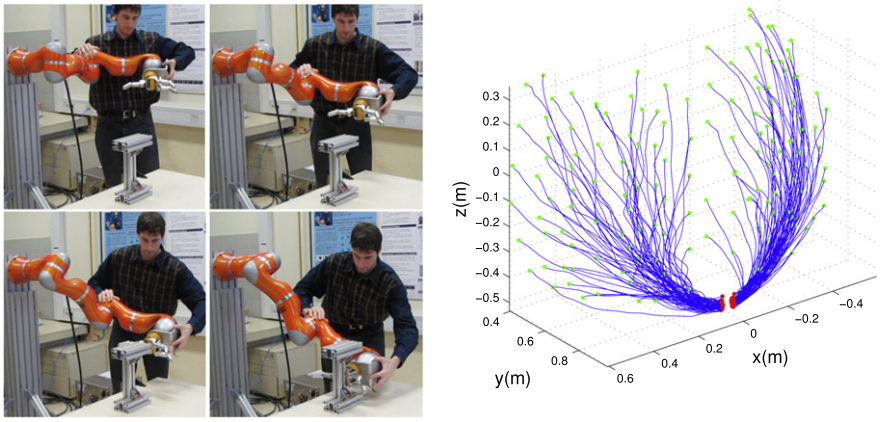


Fig. 14. Pictures show the acquisition of reaching movements by kinesthetic guiding of KUKA Light-Weight Robot arm. The graph shows 144 reaching movements that were performed, 72 from the right side and 72 from the left side of the object. The starting positions on the trajectories are shown as black dots (green in the web version) and the final positions as grey (red in the web version) dots.

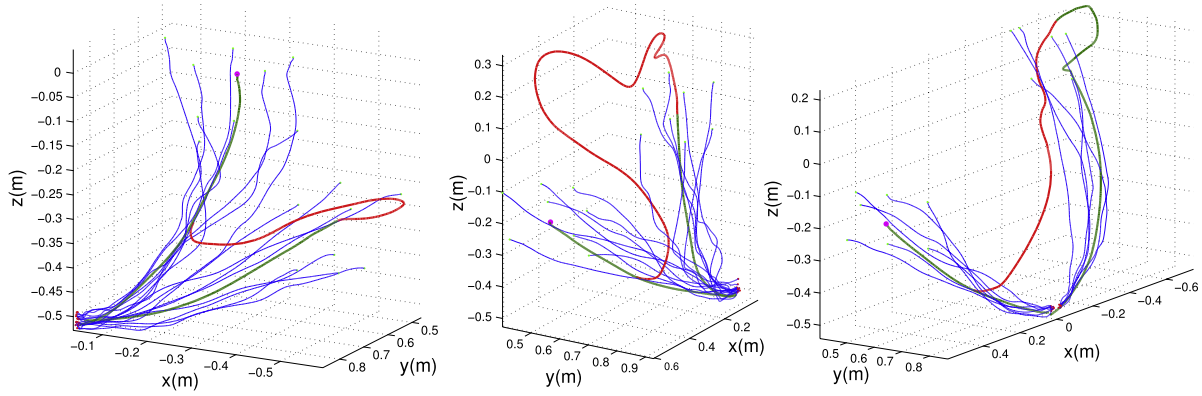


Fig. 15. Reaching and grasping under physical disturbances as performed by Kuka Light-Weight Robot arm. Green curves are the generalized trajectories while blue curves are the nearby demonstrated trajectories. Red curves show the robot motion under physical disturbance. In all graphs the initial, generalized reaching movement encounters a physical disturbance and is pushed to a different position from where the new generalized movement is started. Graphs also show how the initial reaching movements would look like if there were no physical disturbances. The rightmost graph shows how the robot reshapes its initial joint space configuration (the beginning of the second green curve after perturbation) when the movement primitive changes. In this way the generalized trajectory becomes more similar to the training trajectories. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

training trajectories in Cartesian space were used as query points. Cartesian space position of the end-effector determines whether the robot is on the left or right side of the object. Based on this we select the appropriate set of training movements to generate a successful grasping trajectory. We show the performance of real-time generalization in a task in which the robot switches between two different types of trajectories (for left- and right-side grasps) in the case of perturbations.

The KUKA arm was controlled in stiffness mode. While reaching the stiffness is high enough to properly perform the generalized reaching movement. The external joint torques are monitored during execution. If joint torques exceed a threshold (determined empirically), the algorithm switches to a lower stiffness mode, knowing that a physical disturbance occurred. During low stiffness mode the robot is compliant enough to move in the direction of perturbation. Meanwhile, new generalized reaching movements are constantly calculated (every 0.03 s) based on the current position of the robot's end-effector. When the perturbation stops, the newest generalized reaching movement starts being executed. If for example the robot starts reaching from the right side of the object and the perturbation causes it to move to the left side, the algorithm switches from right- to left-side reaching movement. The object is grasped once the robot reaches the end-position on the reaching trajectory. We used a BarrettHand BH-8 Series attached at the top of the arm for grasping. The described reaching and grasping behavior is shown in Figs. 15–17.

4. Conclusion

We developed a new approach for on-line generalization of discrete movements based on Gaussian process regression. The proposed methodology was inspired by motor tape theories [22] and motor schemas [23], in which example movement trajectories are stored directly in memory [24]. Unlike previous generalization approaches, which either required significant on-line calculations [5] or global optimization [2] prone to local minima, the proposed approach can avoid both. Our experiments have shown that despite significant data reduction, which provides the basis for a real-time implementation of the proposed methodology, the generated movements remain close to the ideal movements. The real-time implementation enabled us to realize tasks such as on-line switching between movement primitives based on perceptual feedback, which would not be possible with previous memory-based approaches.

Acknowledgments

This work was supported in part by the European Union Cognitive Systems Project Xperience under Grant FP7-ICT-2009-6-270273, Slovenian Research Agency grant J2-2348, "Brain Machine

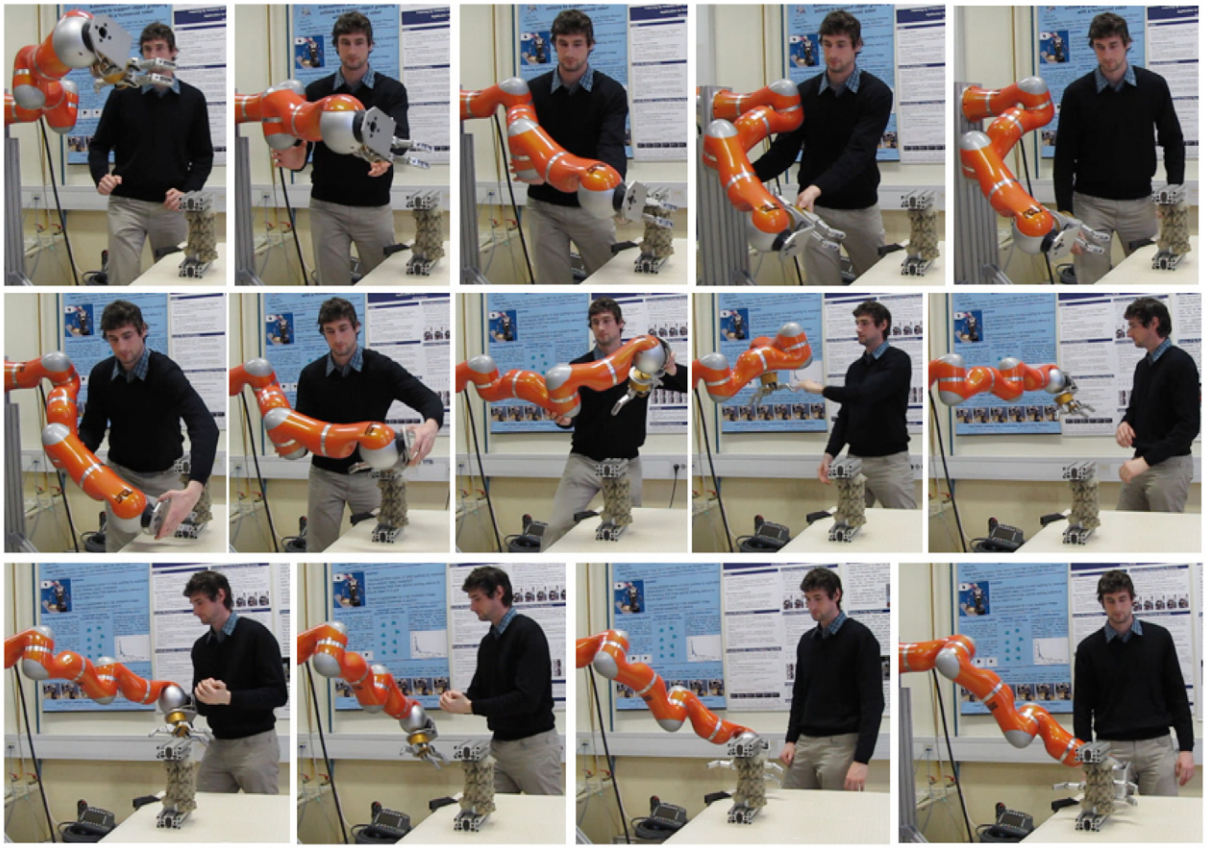


Fig. 16. Reaching with Light-Weight Robot arm is by default controlled in high-stiffness mode. The joint torques are monitored during execution and if a physical disturbance occurs, the algorithm switches to the low stiffness mode where the robot is compliant enough to be pushed to any position from where the new generalized reaching movement is started.

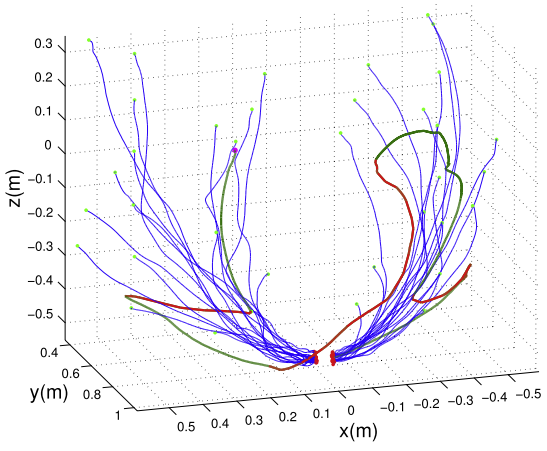


Fig. 17. The 3-D graph illustrates the motion of the Light-Weight Robot arm shown in Fig. 16. Green curves show the generalized trajectories, while blue curves are the nearby demonstrated trajectories. Red curves depict the perturbed robot motion. The beginning of the third green curve shows how the robot reshapes its initial joint configuration when switching to a new movement primitive. This is not necessary if the movement primitive does not change.(For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Interface Development", SBRPS, MEXT, and Grant-in-Aid for Scientific Research on Innovative Areas: Prediction and Decision Making 23120004.

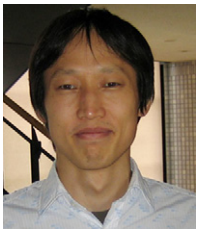
References

[1] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell, A. Billard, Learning and reproduction of gestures by imitation: an approach based on hidden Markov model and Gaussian mixture regression, IEEE Robotics & Automation Magazine 7 (2) (2010) 44–54.
[2] E. Gribovskaya, S.M. Khansari-Zadeh, A. Billard, Learning non-linear multivariate dynamics of motion in robotic manipulators, The International Journal of Robotics Research 30 (1) (2011) 80–117.
[3] V. Krüger, D. Herzog, S. Baby, A. Ude, D. Kragic, Learning actions from observations, IEEE Robotics and Automation Magazine 17 (2) (2010) 30–43.
[4] C. Liu, C.G. Atkeson, Standing balance control using a trajectory library, in: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2009, pp. 3031–3036.
[5] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, IEEE Transactions on Robotics and Automation 26 (5) (2010) 800–815.
[6] D. Bentivegna, C.G. Atkeson, G. Cheng, Learning tasks from observation and practice, Robotics and Autonomous Systems 47 (2–3) (2004) 163–169.
[7] R. Dillmann, Teaching and learning of robot tasks via observation of human performance, Robotics and Autonomous Systems 47 (2–3) (2004) 109–116.
[8] D. Kulić, W. Takano, Y. Nakamura, Online segmentation and clustering from continuous observation of whole body motions, IEEE Transactions on Robotics and Automation 25 (5) (2009) 1158–1166.
[9] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning, Artificial Intelligence Review 11 (1997) 11–73.
[10] C.E. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, MIT Press, Cambridge, MA, 2006.
[11] A.J. Ijspeert, J. Nakanishi, S. Schaal, Learning rhythmic movements by demonstration using nonlinear oscillators, in: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, Lausanne, Switzerland, 2002, pp. 958–963.
[12] A.J. Ijspeert, J. Nakanishi, S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, in: Proc. IEEE Int. Conf. Robotics and Automation, Washington, DC, 2002, pp. 1398–1403.
[13] S. Schaal, P. Mohajerian, A. Ijspeert, Dynamics systems vs. optimal control—a unifying view, Progress in Brain Research 165 (6) (2007) 425–445.

[14] P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, Learning and generalization of motor skills by learning from demonstration, in: Proc. IEEE Int. Conf. Robotics and Automation, Kobe, Japan, 2009, pp. 763–769.
[15] M. Hersch, F. Guenter, S. Calinon, A. Billard, Dynamical system modulation for robot learning via kinesthetic demonstrations, IEEE Transactions on Robotics and Automation 24 (6) (2008) 1463–1467.
[16] A. Ude, C.G. Atkeson, M. Riley, Programming full-body movements for humanoid robots by observation, Robotics and Autonomous Systems 47 (2–3) (2004) 93–108.
[17] D. Nguyen-Tuong, M. Seeger, J. Peters, Model learning with local Gaussian process regression, Advanced Robotics 23 (2009) 2015–2034.
[18] T. Flash, N. Hogan, The coordination of arm movements: an experimentally confirmed mathematical model, The Journal of Neuroscience 5 (7) (1985) 1688–1703.
[19] S. Schaal, C.G. Atkeson, Constructive incremental learning from only local information, Neural Computation 10 (8) (1998) 2047–2084.
[20] A. Ude, E. Oztop, Active 3-D vision on a humanoid head, in: Proc. 14th Int. Conf. Advanced Robotics, Munich, Germany, 2009.
[21] O. Stasse, B. Verrelst, A. Davison, N. Mansard, F. Said, B. Vanderborght, C. Esteves, K. Yokoi, Integrating walking and vision to increase humanoid autonomy, International Journal of Humanoid Robotics 5 (2) (2008) 287–310.
[22] C.G. Atkeson, J. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaal, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, M. Kawato, Using humanoid robots to study human behavior, IEEE Intelligent Systems 15 (4) (2000) 46–56.
[23] R.A. Schmidt, A schema theory of discrete motor skill learning, Psychological Review 82 (1975) 225–260.
[24] T. Poggio, E. Bizzi, Generalization in vision and motor control, Nature 431 (2004) 768–774.

Denis Forte received the Diploma degree from the University of Ljubljana, Ljubljana, Slovenia, in 2009. He is currently a Ph.D. student with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He was a Visiting Researcher at the Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International, Kyoto, Japan, in summer 2010. His research interests include robot learning by statistical methods and automatic feature selection.

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. He was a Visiting Researcher at the Computational Neuroscience Laboratories, Advanced Telecommunications Research Institute International, Kyoto, Japan, in summer 2009. His research interests include human arm motion, learning by imitation and imitation of rhythmic tasks.



Jun Morimoto received the Ph.D. degree in information science from the 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.

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 the 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.