Robotica (2012) volume 30, pp. 837–846. © Cambridge University Press 2011
doi:10.1017/S0263574711001056

Action sequencing using dynamic movement primitives

Bojan Nemec* and Aleš Ude

Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
E-mail: ales.ude@ijs.si

(Accepted August 25, 2011. First published online: October 5, 2011)

SUMMARY

General-purpose autonomous robots must have the ability to combine the available sensorimotor knowledge in order to solve more complex tasks. Such knowledge is often given in the form of movement primitives. In this paper, we investigate the problem of sequencing of movement primitives. We selected nonlinear dynamic systems as the underlying sensorimotor representation because they provide a powerful machinery for the specification of primitive movements. We propose two new methodologies which both ensure that consecutive movement primitives are joined together in a continuous way (up to second-order derivatives). The first is based on proper initialization of the third-order dynamic motion primitives and the second uses online Gaussian kernel functions modification of the second-order dynamic motion primitives. Both methodologies were validated by simulation and by experimentally using a Mitsubishi PA-10 articulated robot arm. Experiments comprehend pouring, table wiping, and carrying a glass of liquid.

KEYWORDS: Trajectory generation; Dynamic movement primitives; Imitation learning.

1. Introduction

Complex tasks are often easier to solve if the overall problem is segmented into parts that are related to subgoals of the task. Such a methodology was used, for example, in the work of Bentivegna *et al.*[1] to implement complex behaviors based on learning from demonstration and practicing[2]. Once the task has been divided successfully into simpler subproblems that a robot can solve efficiently, it is necessary to join the resulting control policies into a continuous motor plan. In robotics as well as in biological systems, motor primitives can be viewed as the building blocks used to construct the entire motor repertoire through different combination operators[3,4]. An increasingly popular representation of motor primitives is based on nonlinear dynamic systems proposed by Ijspeert *et al*[5,6]. The control policies encoded by nonlinear dynamic systems are often referred to as dynamic movement primitives (DMPs). The power of the DMP representation is due to their ability to adapt the learned movements and due to their robustness against perturbations. They were used in many robotics applications comprising learning from demonstration[7–9], skill learning[10,11], movement reproduction with obstacle avoidance[12,13], etc. The effective sequencing of motor primitives can be realized if the robot can smoothly transition from one movement primitive to another. The basic DMP formulation provides enough degrees of freedom only to ensure continuity in positions and velocities. However, when DMPs are used in conjunction with an inverse model controller, it is important that we provide also a continuous desired acceleration signal[8]. This paper contributes to the problem of joining movement primitives by 1) applying 3rd order DMPs with proper initialization of the system variables and 2) with kernel function weights adaptation applied to the 2nd order DMPs. We propose a formulation of dynamic systems that provides enough degrees of freedom to ensure smoothness in position, velocity, and acceleration, which results in smoother transitions between two consecutive motor plans. We address also the problem of combining discrete and rhythmic tasks. In the next section, we briefly describe the standard second-order DMP formulation and introduce the notation, which is used in the rest of the paper. The problem of joining two DMPs is discussed in Section 3, where a solution using third-order DMPs is proposed. The appropriate learning method is presented in Section 4. Next, we propose a method to smoothly join two trajectories based on standard second-order DMP formulation, which uses the technique of kernel function weights adaptation. Experimental results for joining two discrete motions and a discrete and a rhythmic motion are given in Section 6. Finally, we conclude the paper by discussing the properties of our approach in Section 7.

2. Second-Order DMP Formulation

In the DMP framework, any movement can be represented as a time evolution of a nonlinear dynamical system. Such a representation has an advantage that a perturbation can be automatically corrected for by the dynamics of the system, thus enhancing the robustness of the system. In the case of discrete movements, the nonlinear dynamical system can be adapted to new goal configurations by simply changing the goal parameter. Similarly, the DMP can adapt to a new trajectory duration by changing a single parameter. In the standard DMP formulation, motion in each motor, joint, or task coordinate is represented as a damped mass–spring system perturbed by an external force. Such a system can be

* Corresponding author. E-mail: bojan.nemec@ijs.si

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

modeled with a set of differential equations[12,14]

$$\begin{aligned} \dot{v} &= \frac{1}{\tau}\left(K(g-y) - Dv + f(x)\right), \\ \dot{y} &= \frac{v}{\tau}, \end{aligned} \tag{1}$$

where v/τ and y are the velocity and position of the system, x is the phase variable, which defines the time evolution of the trajectory, τ is the temporal scaling factor, K is the spring constant, and D is the damping. For trajectory generation, it is necessary that the dynamic system be critically damped and thus reach the goal position without overshoots. A suitable choice is $D = 2\sqrt{K}$, where K is chosen to meet the desired velocity response of the system. Function $f(x)$ is a nonlinear function which is used to adapt the response of the dynamic system to an arbitrary complex movement. A suitable choice for $f(x)$ was proposed by Ijspeert *et al.*[6] in the form of a linear combination of M radial basis functions

$$f(x) = \frac{\sum_{j=1}^{M} w_j \psi_j(x)}{\sum_{j=1}^{M} \psi_j(x)} x, \tag{2}$$

where ψ_j are Gaussian functions defined as

$$\psi_j(x) = \exp\left(-\frac{1}{2\sigma_j^2}(x - c_j)^2\right). \tag{3}$$

Parameters c_j and σ_j define the center and the width of the jth basis function, while w_j are the adjustable weights used to obtain the desired shape of the trajectory. The phase variable x is defined by

$$\dot{x} = -\frac{\alpha x}{\tau}, \tag{4}$$

where α is an appropriately chosen decay factor. The initial value of the phase variable is 1 and is reset at the beginning of each DMP. In typical robotic application, each DMP corresponds to one controlled variable, which might be, for example, one of the joint coordinates or one of the Cartesian coordinates and all DMPs share the same phase variable.

3. Third-Order DMP Formulation

In the original DMP formulation[6], the system (1)–(4) has an initial state $(x, y, v) = (1, y_0, 0)$ and a unique attractor point $(x, y, v) = (0, g, 0)$, which means that the previous motion has to completely stop before the next motion is generated. Pastor *et al.*[9] noted that by appropriately defining the initial conditions for the second movement, two consecutive movements can be joined together with continuous velocities. In order to obtain the position and velocity continuity, the condition $v_{pred}(x_e)/\tau_{pred} = v_{succ}(1)/\tau_{succ}$ and $y_{pred}(x_e) = y_{succ}(1)$ has to be fulfilled. Suffix *pred* and *succ* denote predecessor and successive motion primitives, respectively, and x_e denotes the phase variable of the predecessor motion primitive at the join instance. The acceleration, however, remains discontinuous even after this modification. In ref. [15], we proposed a solution to this problem by replacing the second-order system (1) with a third-order one, where we added a first-order filter to the DMP output. The third-order DMP was proposed by Schaal *et al.*,[8] where they introduced a simple first-order filter for the goal state in order to ensure continuous accelerations at the goal change. For our purpose, it is worth knowing that any third-order system also allows to control the initial accelerations with proper initialization of the system variables. The third-order canonical system is

$$\begin{aligned} \dot{v} &= \frac{1}{\tau}\left(K(r-y) - Dv + f(x)\right), \\ \dot{y} &= \frac{v}{\tau}, \\ \dot{r} &= \frac{H}{\tau}(g - r). \end{aligned} \tag{5}$$

Schaal *et al.*[8] proposed triple pole of the third-order system, which ensures critical damping of the system. In such a case, gains K, D, and H are related through $D = 2\sqrt{K}$ and $H = \sqrt{K}$. When joining two arbitrary trajectories, one has to assure continuous accelerations and velocities. Initial conditions that ensure continuity of movement up to second-order derivatives can be calculated as follows:

$$\begin{aligned} y_{succ}(1) &= y_{pred}(x_e), \\ v_{succ}(1) &= \dot{y}_{pred}(x_e)\tau_{succ}, \\ r_{succ}(1) &= \frac{\tau_{succ}^2 \ddot{y}_{pred}(x_e) + D\tau_{succ}\dot{y}_{pred}(x_e) - f(1))}{K} \\ &\quad + y_{pred}(x_e). \end{aligned} \tag{6}$$

The phase variable x remains defined by Eq. (4). The proof is as follows. If we omit the nonlinear term f from Eq. (5), equation system (5) becomes a system of nonhomogeneous linear differential equations with constant coefficients. It is well known that the general solution of such a system can be written as a sum of the particular and homogeneous solution, $[z, y]^T = [z_p, y_p]^T + [z_h, y_h]^T$. It is easy to check that a constant function $[0, g, g]$ solves the equation system (5) with the nonlinear term f omitted. Hence, the general solution of this system is given by

$$\begin{bmatrix} v \\ y \\ r \end{bmatrix} = \begin{bmatrix} 0 \\ g \\ g \end{bmatrix} + \exp(t\mathbf{A})\,\mathbf{c}, \quad \mathbf{A} = \begin{bmatrix} -\frac{D}{\tau} & -\frac{K}{\tau} & \frac{K}{\tau} \\ \frac{1}{\tau} & 0 & 0 \\ 0 & 0 & -\frac{H}{\tau} \end{bmatrix}, \tag{7}$$

where $\mathbf{c} \in \mathbb{R}^3$ is an arbitrary constant, which can be determined from the initial conditions. The system is guaranteed to converge to the unique attractor point $[0, g, g]$ if the eigenvalues of $\mathbf{A}$ are negative. The eigenvalues of $\mathbf{A}$ are given as solutions of the equation

$$\det(\mathbf{A} - \lambda\mathbf{I}) = -\left(\lambda^2 + \lambda D/\tau + K/\tau^2\right)(H/\tau + \lambda) = 0, \tag{8}$$

and hence $\mathbf{A}$ has negative eigenvalues $\lambda_{1,2} = -\sqrt{K}/\tau$ and $\lambda_3 = -H/\tau$ if $D = 2\sqrt{K}$, $H, K, \tau > 0$. Since the phase x

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

and consequently the nonlinear term $f(x)$ as defined in (2) tend to zero, the nonlinear system (5) is also guaranteed to converge to the attractor point $[0, g, g]$. The actual value for the third pole λ_3 is selected upon the desired behavior of the system subjected to the goal change. Values of λ_3 closer to 0 increase the smoothing of the trajectory subjected to the goal change during the DMP execution.

An important advantage of DMPs is the ability to change the goal. The standard DMP is defined as a second-order system with three variable inputs: goal g, time constant τ, and the nonlinear function $f(x)$. Whenever a new goal is specified during the DMP evolution, the resulting trajectory is governed according to the response of the second-order system to a step function. An example of such motion is the catching of a free flying object, where the catching position is predicted based on the object position estimates provided by a stereo image system[16], which is changing during the DMP evolution. Thus, the resulting acceleration when applying the second-order DMP is not differentiable, which is not ideal for motor control. On the other hand, the third-order DMP formulation assures smooth differentiable accelerations.

The presented formulation is valid for discrete movements, i.e., movements with output q evolving toward the goal g. Many of the motion patterns in nature are rhythmic, with the motion pattern reproduced cyclically[17]. Rhythmic motions can be described with the same set of equations as discrete motions, which is given in Eq. (1), but with a modified definition of the nonlinear function $f(x)$. The phase variable x, which is the output of the canonical system described by Eq. (4) for discrete motions, is replaced with the phase variable ϕ, which is a monotonically increasing ramp function. The function $f(\phi)$ then becomes

$$f(\phi) = \frac{\sum_{j=1}^{M} w_j \psi_j(\phi)}{\sum_{j=1}^{M} \psi_j(\phi)}, \tag{9}$$

where ψ_j are Gaussian-like periodic functions defined by

$$\psi_j(\phi) = \exp\left(\frac{1}{2\sigma_j^2}(cos(\phi - c_j) - 1\right), \tag{10}$$

and the phase variable ϕ is

$$\tau\dot{\phi} = 1. \tag{11}$$

Parameters c_j and σ_j, respectively, define the center and width of the jth periodic basis function, while w_j are adjustable weights used to obtain the desired shape of the trajectory. The extension to third-order DMPs is trivial. We simply need to replace $f(x)$ with $f(\phi)$ in the set of equations (5).

We tested the sequencing of two discrete movements encoded by DMPs in simulation. In this example, we join two line segments. For the sake of simplicity, the most trivial case of DMP—the line—is used in order to demonstrate solely the effect of discontinuous accelerations when joining DMPs. Joining of more complex trajectories is shown in Section 6. Figure 1 shows the response of two consecutive second- and third-order DMPs and the corresponding velocities

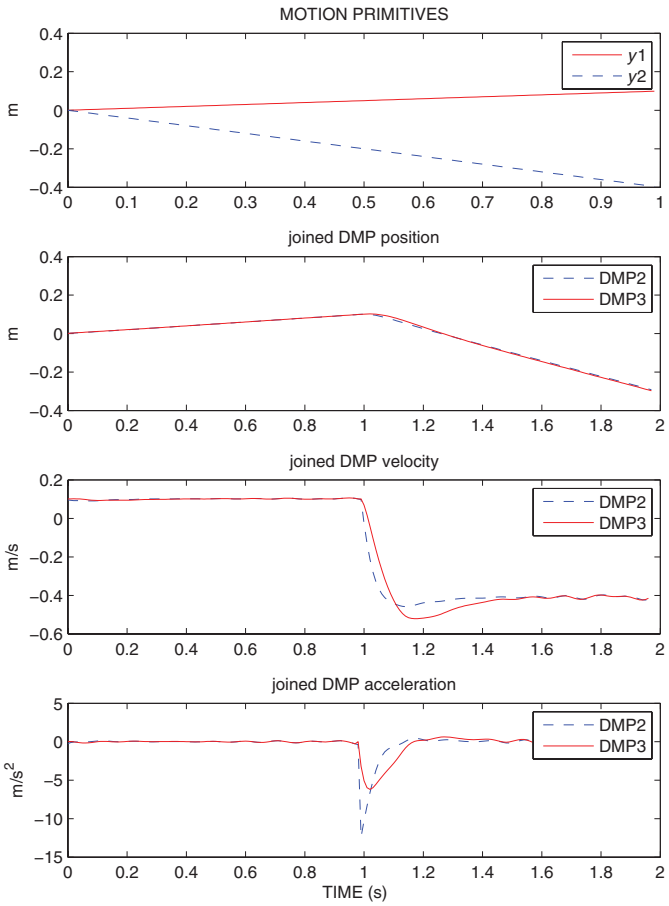

Fig. 1. (Colour online) Sequencing of dynamic movement primitives: the output of the second- and third-order DMP. Panel 1 shows output of each motion primitive, panel 2 shows joined position, panel 3 shows resulting velocity of joined motion primitives, and panel 4 shows resulting accelerations of the joined motion primitives. Labels DMP2 and DMP3 denote second- and third-order DMPs, respectively.

and accelerations. In order to assure smooth transition between two movement primitives, we calculated the initial state of the succeeding movement primitive by utilizing the final state of the preceding movement primitives by exploiting the set of equations (6). As we can see, the third-order DMP formulation ensures continuous accelerations, whereas the accelerations in the original formulation are discontinuous. The parameter values for DMP used in this and the following cases are summarized in Table I. In our experiments, we minimized the number of kernel functions M. We have chosen the number of kernel functions M which still enables the reproduction of the demonstrated trajectory within the desired precision, and optimized the kernel width σ accordingly. More details on optimizing the kernel function number and kernel widths can be found in refs. [17, 18].

4. Motion Acquisition

A trajectory represented by the third-order dynamic system is parameterized with the initial acceleration, velocity, and position, the final goal position, and a set of weights w_j associated with radial basis functions. In this section, we present the procedure for the calculation of

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

Table I. DMP parameters used for the simulation and real robot experiments.

Parameter name	Discrete Motion	Rhythmic Motion
K	200	200
D	28	28
H	14	14
α	2	2
No. of kernels M	10	20
Kernel width σ	0.5	0.32
Integration step dt	0.01 s	0.01 s

weights w_j. We assume that from human demonstration or kinesthetic guiding we obtain trajectory data points $\{y_d(t_i), \dot{y}_d(t_i), \ddot{y}_d(t_i),\ t_i \in [0, T]$. We define the function f^* as follows:

$$f^*(t) = \tau^2 \ddot{y} + \tau D \dot{y} + K(y - g). \tag{12}$$

This function is obtained by replacing the system of two first-order equations (5) with one equation of the second order, where the nonlinear term $f(x)$ has been omitted. Our task is to find a set of weights $\{w_j\}$ that minimize the quadratic cost function

$$J = \sum_{i=0}^{N} (f^*(t_i) - f(x(t_i)))^2. \tag{13}$$

We use global regression methods to find the optimal weights w_j. One possibility is to apply locally weighted regression[6,9], which minimizes M separate cost functions

$$J_j = \sum_{i=0}^{N} \psi_j(x(t_i))(f^*(t_i) - w_j x(t_i))^2, \tag{14}$$

$j = 1, \ldots, M$. Locally weighted regression[19] was proposed as a method that prevents negative interference between task models. Local models are used to generalize in the neighborhood of the given data point and are favorable in conjunction with learning and classification and can help avoid problems with overfitting. The other possibility, which was used in our experiments, is to directly optimize (13), which takes into account the interplay between the neighboring basis functions and can therefore approximate the observed movement with fewer kernel functions and with greater accuracy. Global regression results in the following linear system of equations:

$$\mathbf{A}\mathbf{w} = \mathbf{f}^*, \tag{15}$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix}, \quad \mathbf{f}^* = \begin{bmatrix} f^*(t_0) \\ \vdots \\ f^*(t_N) \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} \frac{\psi_1(x(t_0))x(t_0)}{\sum_{j=1}^{M}\psi_j(x(t_0))} & \cdots & \frac{\psi_M(x(t_0))x(t_0)}{\sum_{j=1}^{M}\psi_j(x(t_0))} \\ \vdots & \vdots & \vdots \\ \frac{\psi_1(x(t_N))x(t_N)}{\sum_{j=1}^{M}\psi_j(x(t_N))} & \cdots & \frac{\psi_M(x(t_N))x(t_N)}{\sum_{j=1}^{M}\psi_j(x(t_N))} \end{bmatrix}.$$

Similarly, as in the case of locally weighted regression, it is possible to compute a solution to (15) recursively by incrementally updating the following quantities:

$$\mathbf{P}_i = \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{a}_i\mathbf{a}_i^T\mathbf{P}_{i-1}}{1 + \mathbf{a}_i^T\mathbf{P}_{i-1}\mathbf{a}_i}, \tag{16}$$

$$\mathbf{w}_i = \mathbf{w}_{i-1} + (f^*(t_i) - \mathbf{a}_i^T\mathbf{w}_{i-1})\mathbf{P}_i\mathbf{a}_i, \tag{17}$$

where $\mathbf{a}_i$ is the M-dimensional column vector associated with the corresponding row of the matrix $\mathbf{A}$ and the optimal weights are $\mathbf{w} = \mathbf{w}_N$. When learning third-order DMPs, we set $r = g$. With this setting for r and because there are no changes in the goal configuration g, the third-order DMP becomes equal to the second-order DMP and we can use the same training method as for second-order DMPs. In contrast to this approach, the third-order formulation presented in ref. [15] also requires the jerk estimation.

5. Online Kernel Function Adaptation

To avoid increasing the order of the underlying system of differential equations, we also explored a solution to motion sequencing, which is based on online modification of the Gaussian kernel functions gains. This second approach enables us to keep using the original DMP formulation. The goal of this modification is to assure continuous acceleration when joining two motion primitives. We assume that we join two trajectories at time t_0. The phase variable x of the succeeding DMP is 1 for $t = t_0$ and the preceding DMP ended with phase variable x_e at the time $t_0 - dt$, where dt denotes the sampling time. Let's rewrite (1) for $x = 1$, i.e.,

$$\dot{v}(1) = \frac{1}{\tau}\left(K(g - y(1)) - Dv(1) + f(1)\right). \tag{18}$$

Obviously, in order to assure continuity, $\dot{v}_{pred}(x_e)/\tau_{pred} = \dot{v}_{succ}(1)/\tau_{succ}$, $v_{pred}(x_e)/\tau_{pred} = v_{succ}(1)/\tau_{succ}$ and $y_{pred}(x_e) = y_{succ}(1)$, which leads to the condition

$$f_{succ}(1) = \frac{\tau_{succ}^2}{\tau_{pred}}\dot{v}_{pred}(x_e) + \frac{\tau_{succ}}{\tau_{pred}}Dv_{pred}(x_e) - K(g_{succ} - y_{pred}(x_e)). \tag{19}$$

Suffix *pred* and *succ* denote predecessor and successive motion primitives, respectively. From (2), it follows that

$$w_1 = f_{succ}(1) - \frac{\sum_{j=2}^{M} w_j\psi_j(1)}{\sum_{j=1}^{M}\psi_j(1)}, \tag{20}$$

since $\psi_1(1) = 1$ by definition. In order to assure continuous accelerations when joining two dynamic movement

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

primitives, it is necessary to recalculate the weight of the first Gaussian kernel function of the next motion primitive according to Eq. (20).

Modification of just the first Gaussian kernel function might not be appropriate when applying the wider Gaussian kernel function (3). A general solution for such a case yields to

$$\triangle \mathbf{w} = \hat{\boldsymbol{\psi}} \left(f_{succ}(1) \sum_{j=1}^{M} \psi_j(1) - \sum_{j=1}^{M} w_j \psi_j(1) \right), \qquad (21)$$

where $\hat{\boldsymbol{\psi}}$ denotes the Moore–Penrose pseudoinverse of the vector $\boldsymbol{\psi}$ composed of components of the Gaussian kernel functions $\psi_j(1)$ and $\triangle \mathbf{w}$ denotes the vector of modification of the original weights $\mathbf{w}$.

In the case when a discrete motion is followed by a rhythmic motion, we apply a similar strategy of modifying the weight of the appropriate kernel function. In contrast to discrete motions, rhythmic motions can be started at arbitrary phase ϕ. In such a case, the initial acceleration is determined by w_k, where k is the index where the function $\psi_k(\phi)$ is maximal, $k = \{1..N\}$. The weight of the kth kernel function can be calculated by

$$w_k = \left(\frac{\tau_{succ}^2}{\tau_{pred}} \dot{v}_{pred}(x_e) + \frac{\tau_{succ}}{\tau_{pred}} D v_{pred}(x_e) - K(g_{succ} - y_{pred}(x_e)) - \frac{\sum_{j=1, j\neq k}^{M} w_j \psi_j(\phi)}{\sum_{j=1}^{M} \psi_j(\phi)} \right) \frac{1}{\psi_k(\phi)}. \qquad (22)$$

In the above equation, we have two phase variables, x and ϕ, one belonging to the preceding discrete motion and the other to the succeeding rhythmic motion. Note that the modified kernel function w_k should be used only in the transition phase when joining discrete and rhythmic motion. Afterward, the original kernel function weight should be used in order to preserve the shape of the learned rhythmic motion.

In order to demonstrate the efficiency of the proposed modification, we repeated the simulation for the same case as in Section 3. Here, we compared the response of the ordinary second-order DMP and the second-order DMP with the online weights modification according to Eq. (5). From Fig. 2, we can see that the modified DMP results in continuous accelerations when joining two arbitrarily chosen DMPs.

6. Experimental Results

In this section, we experimentally evaluate DMP sequencing algorithms on a real robot. As a representative example that involves the joining of two discrete movements, we consider the pouring task. Let's briefly describe the task of pouring a liquid into a glass. It depends on many factors including the position of the glass with respect to the body, the shape of the vessel containing the liquid, and the shape of the glass to be filled. A general strategy for pouring is 1) approach the glass to be filled with a suitable approach trajectory and 2) start the pouring motion toward the end of the approach

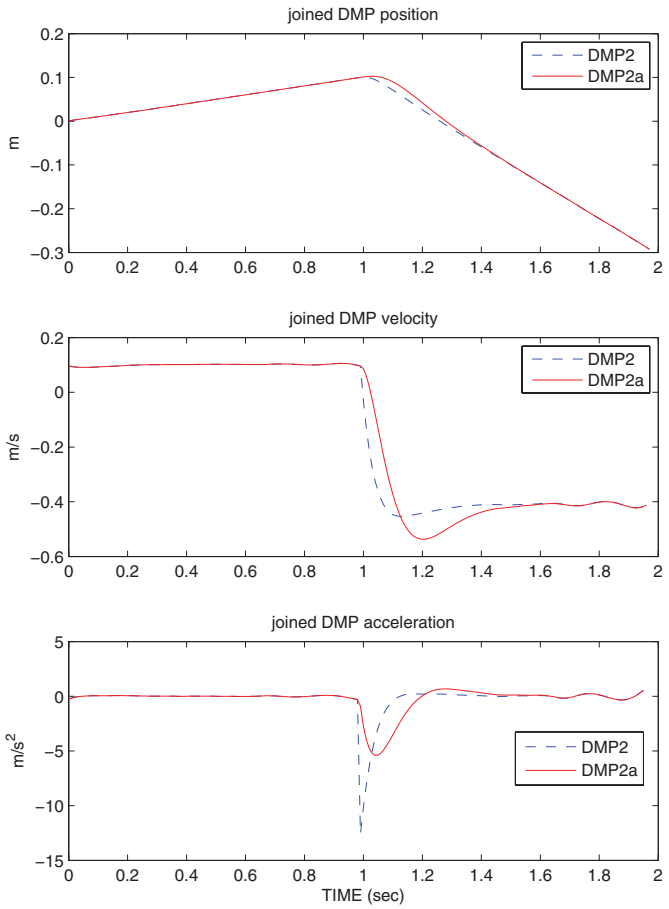

Fig. 2. (Colour online) Sequencing of dynamic movement primitives: the output of the ordinary second-order DMP (labeled as DMP2) and DMP with online weights adaptation (labeled as DMP2a).

trajectory. During the execution of the pouring motion, the robot monitors the liquid flow until the glass is filled to the desired level. These two phases define two separate movement primitives. To successfully fill the glass placed at different locations on the table, the actual pouring motion does not need to be changed. Successful pouring can be achieved solely by selecting the appropriate goal position for the approach trajectory, which is automatically taken into consideration by the underlying dynamic system, followed by the previously learned and constant pouring movement. The goal position of the approach trajectory can be determined with an appropriate sensory system, e.g., vision. For this approach to work, we need to be able to smoothly sequence the available primitives. Figure 3 shows our experimental setup. It consists of the seven degree-of-freedom articulated robotic arm Mitsubishi PA-10 with the three-finger Barrett hand. A 3-D vision subsystem was used to extract features of the scene, i. e., the glass and the vessel position on the table. The task of the robot was to grasp the vessel (bottle) and to fill the glass. In order to accomplish the task, we used movement primitives from our library of DMPs. Our DMP library consists of various motion primitives relevant in typical kitchen tasks: reaching, pouring, wiping, shaking, cutting, power grasps, etc. Motion primitives were obtained from human demonstration using an optical tracking system based on passive markers (BTS Smart[20]). All motion primitives

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

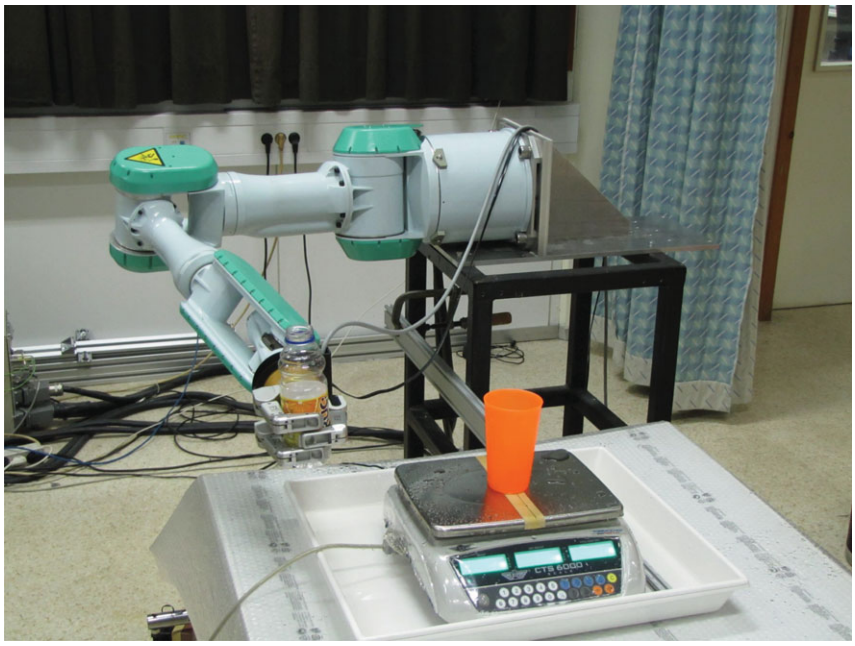

Fig. 3. (Colour online) Experimental setup for pouring.

were encoded as DMPs using the procedure described in Section 4 and are independent regarding the usage of the second- or the third-order DMPs. DMPs were designed to encode discrete point-to-point movements which starts and ends when the robot is at rest, i.e., with zero velocity and acceleration. The procedures described in Sections 3 and 5 enable a transition between two DMPs without requiring that the velocities and accelerations are zero. This is achieved by initializing the initial DMP parameters y and v with values different from what was observed during training. One of the key issues of extracting motion primitives during online observation is also the motion segmentation. Recently, few approaches for automatic motion segmentation were proposed, based on combined stochastic segmentation and clustering[21], using the observation in object space[22], by observing the velocities at zero crossing of the joint motion[23], etc. In our case, the observed motion was segmented manually into appropriate motion primitives.

For the pouring task, the transition between the approach and pouring primitive was accomplished without stopping the motion at the end of each movement. In order to compare methods, we applied ordinary second-order DMPs, DMPs with online weights adaptation, and third-order DMPs. In all cases, we parameterized dynamic movement primitives with 10 kernel functions. The results are presented in Fig. 4. For the sake of simplicity, only the tilt angle is shown. The dotted line in Fig. 4 marks the time when we join both motion primitives. We can see that both proposed methods, the third-order DMP and DMP with online modification of weights of the kernel functions result in continuous accelerations, whereas the original second-order method does not. As expected, the response of the second-order DMP differs from the second-order DMP with weight modifications only at

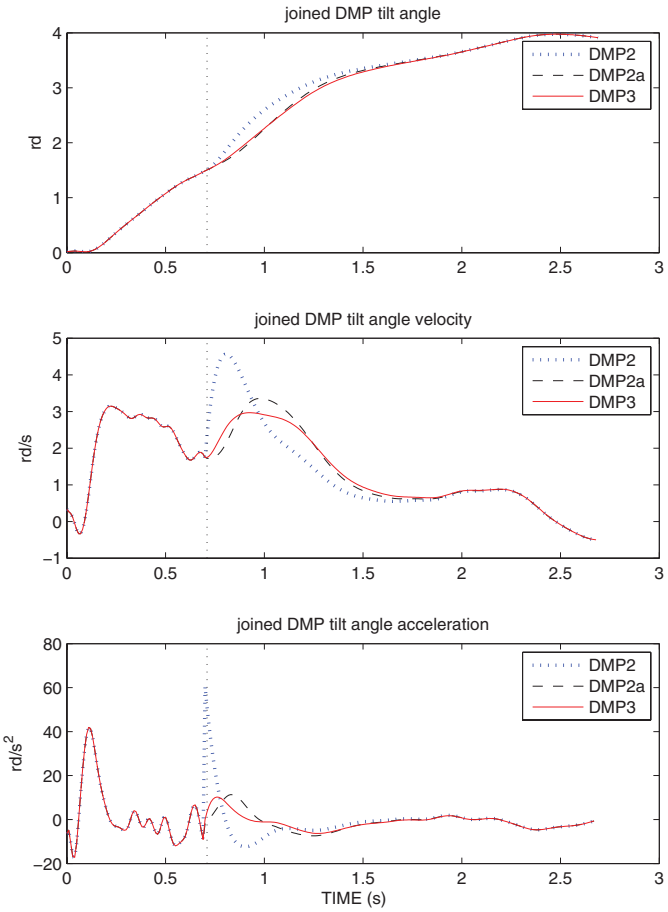

Fig. 4. (Colour online) Sequencing of dynamic movement primitives: the output of the ordinary second-order DMP labeled as DMP2, DMP with online weights adaptation labeled as DMP2a, and third-order DMP labeled as DMP3.

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

Fig. 5. (Colour online) Experimental setup for wiping.

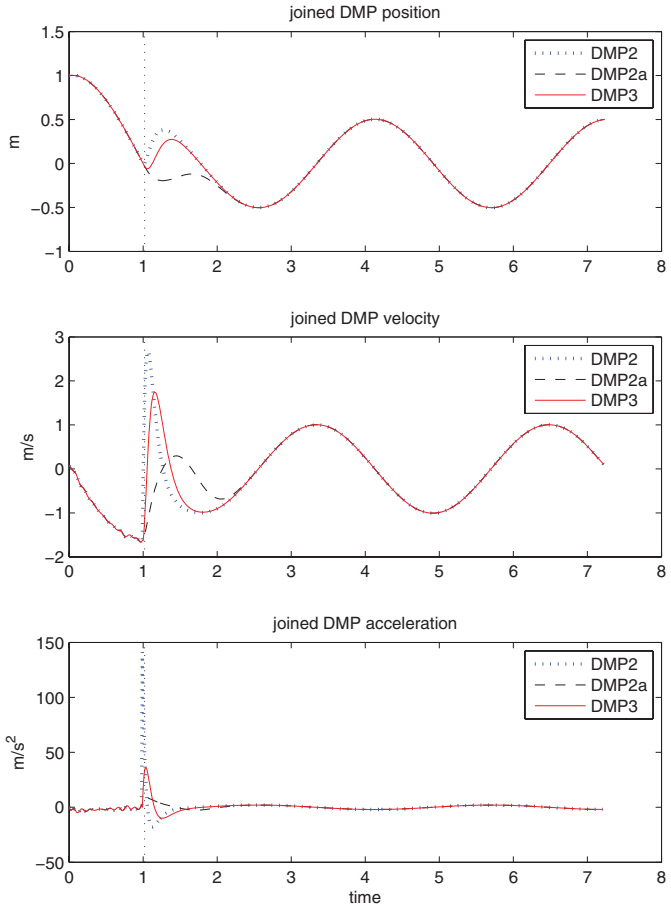

Fig. 6. (Colour online) Sequencing of discrete and rhythmic motions: the output of the ordinary second-order DMP (DMP2), DMP with online weights adaptation (DMP2a), and third-order DMP (DMP3).

the beginning of motion primitive. Later, it converges to the same trajectory as specified by the standard second-order DMP.

Next, we experimentally evaluated the sequencing of discrete and rhythmic motions. Here, we considered the task of wiping the table, as shown in Fig. 5. The initial motion was a discrete movement, which brings the robot to the starting position for wiping. This approach motion was followed by the wiping motion, which has a rhythmic pattern.

The transition between the approach and the wiping motion primitive is accomplished in such a way that we assure continuous velocities and accelerations. As before, we compared the response of the ordinary second-order DMPs, DMPs with online weight adaptation, and third-order DMP. The discrete and rhythmic motion were parameterized with 10 and 20 kernel functions, respectively. The results are presented in Fig 6. For the sake of simplicity, only the x coordinate is shown. As in the previous example, we can see that the proposed approaches with the third-order DMP and the second-order DMP with online adaptation of kernel gains assure smooth transition when joining two trajectories, whereas the original DMP formulation results in discontinuous accelerations.

The third example illustrates the importance of continuous acceleration while executing typical tasks of kitchen scenario. The task of the robot was to move a glass of liquid from the table to a different location. The task was composed of two motion primitives as shown in Fig. 7. Again, we parameterized dynamic movement primitives with 10 kernel functions. Joining ordinary second-order DMPs results in motion with discontinuous accelerations and consequently, the robot spills the liquid as shown in Fig. 8. On the other hand, by using DMPs with online weights adaptation or third-order DMPs with a proper initialization, the robot successfully accomplished the task without spilling the liquid. The reference position, velocity, and acceleration trajectory for the y coordinate of the joined motion is shown in Fig. 9. The effect of discontinuous acceleration becomes even more evident with faster movements.

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

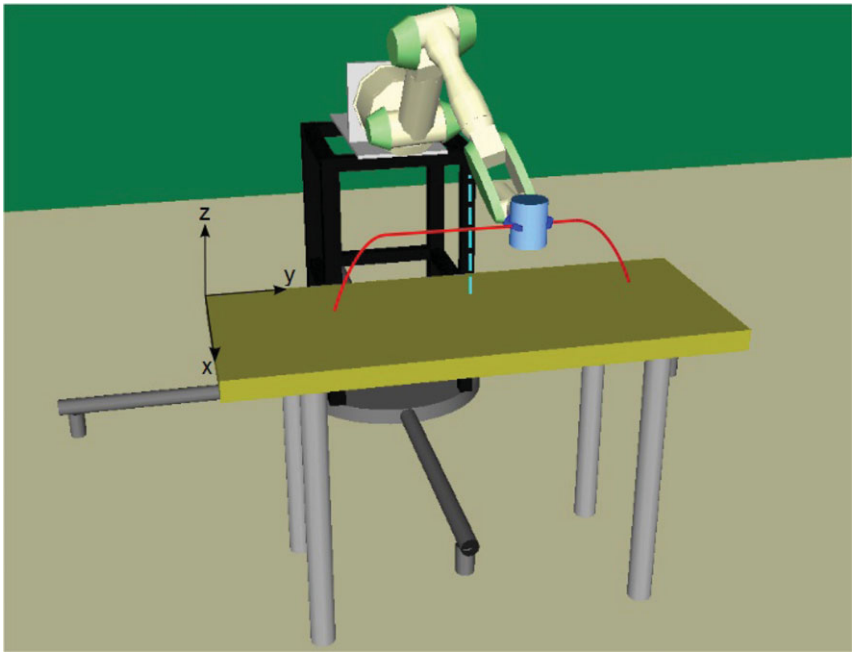

Fig. 7. (Colour online) The task describes pick and place sequence for a glass filled with liquid. Red solid line shows the robot trajectory and the dashed blue line indicates the split point of two DMPs used to accomplish this task.

7. Conclusions

We studied the problem of joining dynamic movement primitives, where we allow for nonzero velocity at the moment of joining two consecutive primitives. With standard, second-order DMPs, it is only possible to specify the initial position and velocity for the following DMP, even with modification proposed in Pastor *et al.*[9]. In general, it is therefore not possible to join two consecutive primitives with continuous acceleration. In order to ensure continuity up to the second-order derivatives, we proposed two approaches. The first is based on the formulation of dynamic movement primitives based on nonlinear third-order dynamical system, where the additional degree of freedom acts as a first-order filter for the desired goal. This approach uses the same third-order dynamical system as proposed by Schaal *et al.*[8]. We propose proper initialization of the system variables in order to control the initial accelerations. A benefit of the third-order DMPs is that we obtain smooth accelerations also in the case when we change the goal position during the execution of the trajectory, which can happen very often when the goal position is provided from a sensory system, e. g., a camera. The second approach is based on online modification of Gaussian kernel functions using the standard second-order DMP formulation. The initial acceleration is influenced mainly by the weight of the first few Gaussian kernel functions. Therefore, we proposed to change them online according to the required initial acceleration when joining two dynamic movement primitives. Generally, both methods perform similarly when joining two consecutive motion primitives with continuous acceleration, although

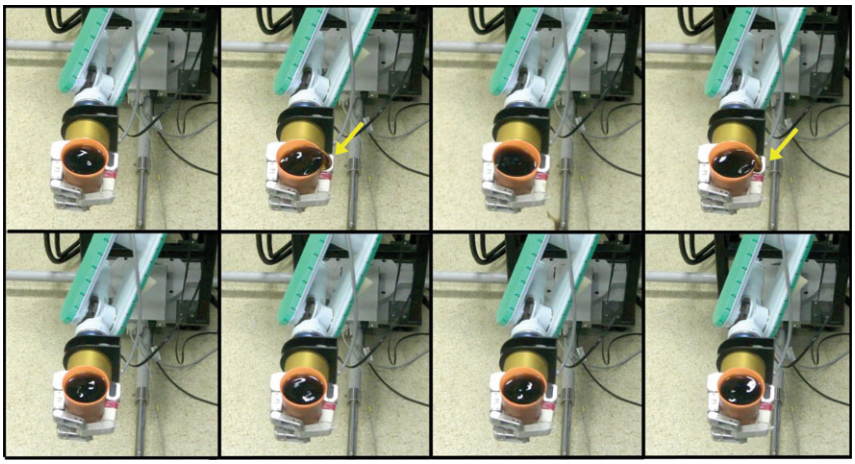

Fig. 8. (Colour online) The upper sequence shows that discontinuous accelerations in the case of joining DMPs with ordinary second-order DMPs result in spilling the liquid from the glass. On the other hand, joining trajectory with third-order DMPs and with second-order DMPs and online kernel adaptation does not cause the liquid spilling, as shown in the lower sequence. There were no visible differences between the method using third-order DMPs and the method using second-order DMPs with the online kernel function adaptation.

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

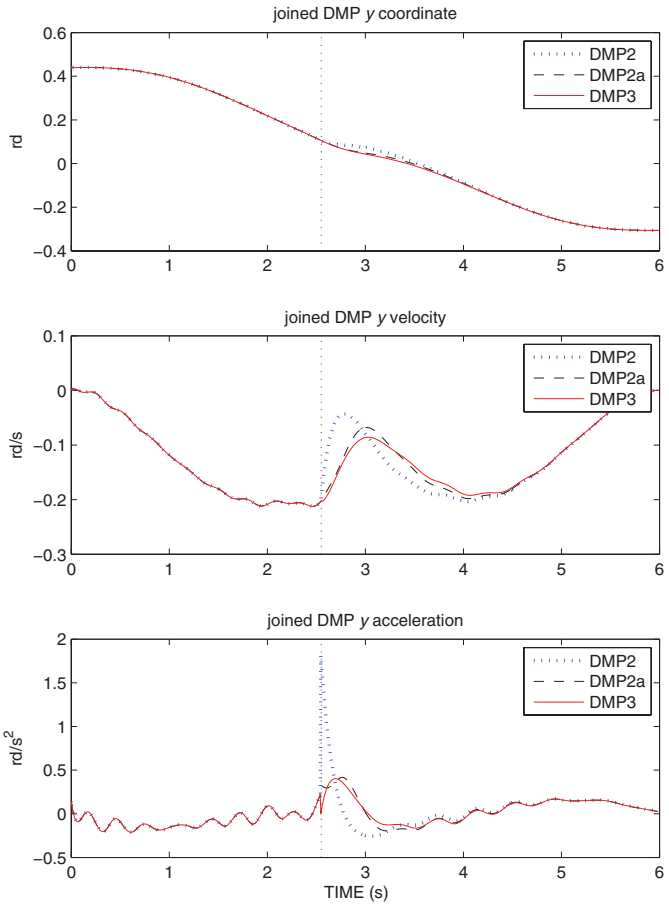

Fig. 9. (Colour online) Sequencing of two discrete motions in the task of carrying the glass: the output of the ordinary second-order DMP (DMP2), second-order DMP with online weights adaptation (DMP2a), and third-order DMP (DMP3).

the second approach can generate smoother acceleration, as shown in Fig. 6. The benefit of the second approach is in its simplicity. Both methods were tested in simulation using a simple illustrative example, where we joined two ramp functions, as well as in real-world experiments that included pouring a liquid into a glass, table wiping, and carrying a glass. The third experiment shows that continuous accelerations are essential when performing typical kitchen scenario tasks such as carrying a glass of liquid, even at relatively low velocities. It has been shown that both proposed approaches are appropriate when joining any combination of discrete and rhythmic motions for the cases where smooth accelerations are important. On the other hand, for some applications, it might even be good to have jumps in the acceleration to react more rapidly. Both methods allow specifying the initial acceleration of the subsequent DMP, therefore the proposed technique could be used also to increase the accelerations in order to handle the cases where fast reactions are more important than smooth trajectories.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 269959, IntellAct. It was also funded in part by the Slovenian Research Agency grant J2-2348.

References

1. D. C. Bentivegna, C. G. Atkeson, A. Ude and G. Cheng, "Learning to act from observation and practice," *Int. J. Humanoid Robot.* **1**(4), 585–611 (2004).
2. R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robot. Auton. Syst.* **47**(2), 109–116 (2008).
3. D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Netw.* **11**, 1317–1329 (1998).
4. M. J. Matarić, *Sensory Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics* (MIT Press, 2002).
5. A. J. Ijspeert, J. Nakanishi and S. Schaal, "Learning Rhythmic Movements by Demonstration Using Nonlinear Oscillators," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland (2002) pp. 958–963.
6. A. J. Ijspeert, J. Nakanishi and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC (2002) pp. 1398–1403.
7. A. J. Ijspeert, J. Nakanishi and S. Schaal, "Learning Attractor Landscapes for Learning Motor Primitives," **In**: *Advances in Neural Information Processing Systems 15* (S. Becker, S. Thrun and K. Obermayer, eds.) (MIT Press, Cambridge, MA, 2003) pp. 1547–1554.
8. S. Schaal, J. Peters, J. Nakanishi and A. Ijspeert, "Learning Movement Primitives," **In**: *Robotics Research: The Eleventh International Symposium* (P. Dario and R. Chatila, eds.) (Springer, Berlin, Heidelberg, 2005) pp. 561–572.
9. P. Pastor, H. Hoffmann, T. Asfour and S. Schaal, "Learning and Generalization of Motor Skills by Learning from Demonstration," *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan (2009) pp. 763–768.
10. J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.* **21**, 682–697 (2008).
11. J. Kober and J. Peters, "Learning Motor Primitives for Robotics," *ICRA'09: Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, Kobe, Japan (2009) pp. 2112–2118.
12. D.-H. Park, H. Hoffmann, P. Pastor and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," *Proceedings of the 8th IEEE-RAS International Conference on Humanoid Robots* (2008) pp. 91–98.
13. H. Hoffmann, P. Pastor, D. Park and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," *Proceedings of the IEEE International Conference on Robotics and Automation*, Kobe, Japan (2009) pp. 2587–2592.
14. S. Schaal, P. Mohajerian and A. Ijspeert, "Dynamics systems vs. optimal control—a unifying view," *Prog. Brain Res.* **165**(6), 425–445 (2007).
15. B. Nemec, M. Tamosiunaite, F. Worgotter and A. Ude, "Task Adaptation through Exploration and Action Sequencing, *Proceedings of the 9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France (2009) pp. 610–616.
16. B. Nemec, M. Zorko and L. Žlajpah, "Learning of a Ball-in-a-Cup Playing Robot," *Proceedings of the 19th IEEE International Workshop on Robotics in Alpe-Adria-Danube Region*, Budapest, Hungary (2010) pp. 297–301.
17. A. Gams, A. Ijspeert, S. Schaal and J. Lenarcic, "On-line learning and modulation of periodic movements with nonlinear dynamical systems," *Auton. Robots* **27**(1), 3–23 (2009).

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press

18. A. Ude, A. Gams, T. Asfour and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robot.* **26**(5), 800–815 (2010).
19. C. G. Atkeson, A. W. Moore and S. Schaal, "Locally weighted learning," *AI Review* **11**, 11–73 (1997).
20. BTS Bioengineering. High frequency digital system for biomechanical motion analysis. http://www.btsbioengineering.com/Media/Brochure/assets/bts_smartd_a40308uk.pdf.
21. D. Kulič, W. Takano and Y. Nakamura, "Online segmentation and clustering from continuous observation of whole body motions," *IEEE Trans. Robot.* **25**(5), 1158–1166 (2009).
22. V. Krueger, L. D. Herzog, S. Baby, A. Ude and D Kragic, "Learning actions from observation," *IEEE Robot. Autom. Mag.* **17**(2), 30–43 (2010).
23. A. Fod, M. J. Matarič and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Auton. Robots* **12**(1), 39–54 (2002).

https://doi.org/10.1017/S0263574711001056 Published online by Cambridge University Press