Control, Planning, Learning, and Imitation with Dynamic Movement Primitives

Stefan SCHAAL[1,2], Jan PETERS[1,2], Jun NAKANISHI[2], and Auke IJSPEERT[1,2,3]

Computational Learning and Motor Control Laboratory[1]
Computer Science and Neuroscience, University of Southern California, Los Angeles, CA 90089-2520, USA
Department of Humanoid Robotics and Computational Neuroscience[2]
ATR Computational Neuroscience Laboratory, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, 619-0288 Kyoto, JAPAN
School of Computer and Communication Sciences[3]
EPFL, Swiss Federal Institute of Technology Lausanne, CH 1015 Lausanne, SWITZERLAND

Abstract. In both human and humanoid movement science, the topic of movement primitives has become central in understanding the generation of complex motion with high degree-of-freedom bodies. A theory of control, planning, learning, and imitation with movement primitives seems to be crucial in order to reduce the search space during motor learning and achieve a large level of autonomy and flexibility in dynamically changing environments. Movement recognition based on the same representations as used for movement generation, i.e., movement primitives, is equally intimately tied into these research questions. This paper discusses a comprehensive framework for motor control with movement primitives using a recently developed theory of dynamic movement primitives (DMP). DMPs are a formulation of movement primitives with autonomous nonlinear differential equations, whose time evolution creates smooth kinematic movement plans. Model-based control theory is used to convert such movement plans into motor commands. By means of coupling terms, on-line modifications can be incorporated into the time evolution of the differential equations, thus providing a rather flexible and reactive framework for motor planning and execution — indeed, DMPs form complete kinematic control policies, not just a particular desired trajectory. The linear parameterization of DMPs lends itself naturally to supervised learning from demonstrations. Moreover, the temporal, scale, and translation invariance of the differential equations with respect to these parameters provides a useful means for movement recognition. A novel reinforcement learning technique based on natural stochastic policy gradients allows a general approach of improving DMPs by trial and error learning with respect to almost arbitrary optimization criteria, including situations with delayed rewards. We demonstrate the different ingredients of the DMP approach in various examples, involving skill learning from demonstration on the humanoid robot DB and an application of learning simulated biped walking from a demonstrated trajectory, including self-improvement of the movement patterns in the spirit of energy efficiency through resonance tuning.

1 Introduction

With the advent of anthropomorphic and humanoid robots [1], a large number of new challenges have been posed to the field of robotics and intelligent systems. Lightweight, highly complex, high degree-of-freedom (DOF) bodies defy accurate analytical modeling such that movement execution requires novel methods of nonlinear control based on learned feedforward controllers[2]. This issue is even amplified since, due to frequent contacts with an unknown environment, high gain control is not a viable alternative to model-based control. Movement planning in high dimensional motor systems offers another challenge. While efficient planning in typical low dimensional industrial robots, usually characterized by three to six DOFs, is already a complex issue[3,4], optimal planning in 30 to 50 DOF systems with uncertain geometric and dynamic models is quite daunting, particularly in the light of the required real-time performance in a reactive robotic system. For example, the field of reinforcement learning[5], one of the most general planning frameworks, has hardly progressed beyond four to six dimensional problems in continuous state and action problems. As one more point, advanced sensing, using vision, tactile sensors, acoustic sensors, and potentially many other sources like olfaction, distributed sensor networks, nanosensors, etc., play a crucial role in advanced robotics. Besides finding reliable methods of processing in such sensor rich environments[6], incorporating the resulting information into the motor and planning loops increases the above mentioned complexity of planning and control even more.

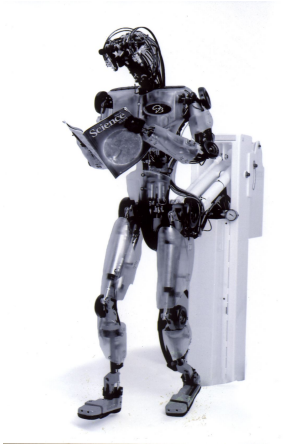

Fig. 1. Humanoid robot DB

From this brief outline of some of the basic problems of advanced robotics, it is apparent that robotics researchers face increasingly more the full complexity of information processing that biology solves so seeming effortlessly on

a daily basis. Thus, increasingly more projects resort to studying human behavior and neurobiological phenomena in order to extract relevant principles for new methods in robotics. Understanding dexterous manipulation skills [7,8], imitation learning [9–12], human movement recognition [13], human active visual perception[14–16], human locomotion[17], and the composition of complex actions[18–20] are among the most salient topics.

One of the fundamental questions, recurring in many of the above lines of research, revolves around identifying movement primitives (a.k.a. units of actions, basis behaviors, motor schemas, etc.)[9,11,21–25]. The existence of movement primitives seems, so far, the only possibility how one could conceive that biological and artificial motor systems can cope with the complexity of motor control and motor learning[9,26,27]. Developing a theory of control, planning, learning, and imitation with movement primitives for humans and humanoids is therefore currently a rather prominent topic in both biological and robotic sciences.

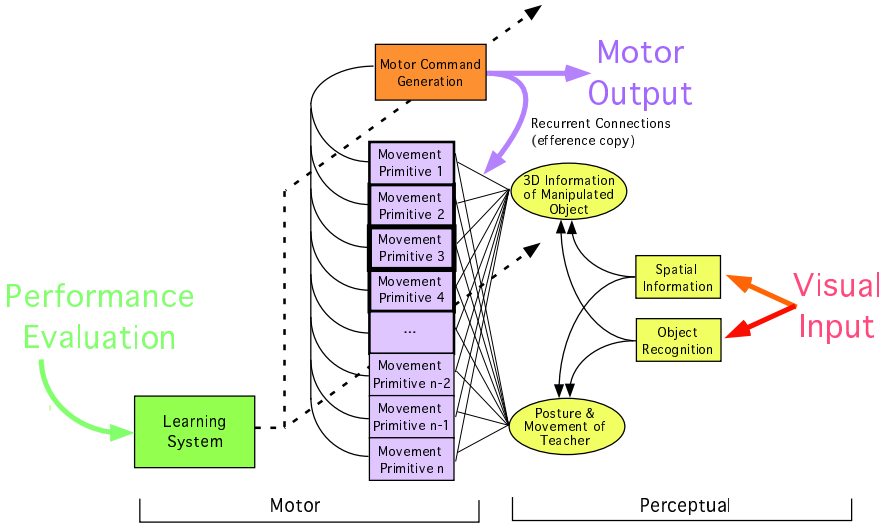

Fig. 2. Conceptual sketch of motor control and motor learning with movement primtiives. The right side of the figure contains primarily perceptual elements and indicates how visual information is transformed into spatial and object information, as needed for supervised learning from demonstration. The left side focuses on motor elements, illustrating how a set of movement primitives competes for a demonstrated behavior, and finally selects the most appropriate one for execution and further refinement through trial and error. Motor commands are generated from input of the most appropriate primitive. Learning can adjust both movement primitives and the motor command generator.

Our approach to motor control with movement primitives, sketched in Figure 2[9], was motivated by the desire to speed up motor learning with imitation learning. It is also intended to provide a framework for movement recognition, a principled way of action generation that allows generalization of a learned movement to related tasks, and to include a means for perception-action coupling as needed in many interactive perceptuomotor skills. In the following sections, we will first sketch our idea of Dynamic Movement Primitives, originally introduced in [28–30], illustrate their potential for imitation learning and general reinforcement learning, and exemplify this framework in various applications from humanoid robotics with the humanoid robot DB (Figure 1) and simulation studies.

2 Control Policies

The goal of motor learning can generally be formulated in terms of finding a task-specific control policy:

$$\mathbf{u} = \pi\left(\mathbf{x}, t, \alpha\right) \tag{1}$$

that maps the continuous state vector $\mathbf{x}$ of a control system and its environment, possibly in a time t dependent way, to a continuous control vector $\mathbf{u}$[31,32]. The parameter vector α denotes the problem specific adjustable parameters in the policy π, e.g., the weights in neural network or a generic statistical function approximator. Given some cost criterion that can evaluate the quality of an action $\mathbf{u}$ in a particular state $\mathbf{x}$, dynamic programming, and especially its modern relative, reinforcement learning, provide a well founded set of algorithms of how to compute the policy π for complex non-linear control problems. Unfortunately, as already noted in Bellman's original work, learning of π becomes computationally intractable for even moderately high dimensional state-action spaces. Although recent developments in reinforcement learning increased the range of complexity that can be dealt with [5,33,34], it still seems that there is a long way to go to apply general policy learning to complex control problems.

In most robotics applications, the full complexity of learning a control policy is strongly reduced by providing prior information about the policy. The most common priors are in terms of a desired trajectory, $[\mathbf{x}_d(t), \dot{\mathbf{x}}_d(t)]$, usually handcrafted by the insights of a human expert. For instance, by using a PD controller, a (explicitly time dependent) control policy can be written as:

$$\begin{aligned}\mathbf{u} &= \pi\left(\mathbf{x}, \alpha\left(t\right), t\right) = \pi\left(\mathbf{x}, [\mathbf{x}_d(t), \dot{\mathbf{x}}_d(t)], t\right) \\ &= \mathbf{K}_x\left(\mathbf{x}_d(t) - \mathbf{x}\right) + \mathbf{K}_{\dot{x}}\left(\dot{\mathbf{x}}_d(t) - \dot{\mathbf{x}}\right)\end{aligned} \tag{2}$$

For problems in which the desired trajectory is easily generated and in which the environment is static or fully predictable, as in many industrial applications, such a shortcut through the problem of policy generation is highly

successful. However, since policies like in (2) are usually valid only in a local vicinity of the time course of the desired trajectory, they are not very flexible. When dealing with a dynamically changing environment in which substantial and reactive modifications of control commands are required, one needs to adjust trajectories appropriately, or even generate entirely new trajectories by generalizing from previously learned knowledge. In certain cases, it is possible to apply scaling laws in time and space to desired trajectories[35,36], but those can provide only limited flexibility. Novel approaches developed in computer graphics advocate to combine trajectory pieces recorded from motion capture to form more complex desired trajectories in a flexible task-specific way[18]. However, it remains a topic of future work whether these methods can be applied to the on-line control of complex robots, e.g., potentially employing methods of dynamics filtering[37] of motion capture data in order to ensure physical realizability for a particular robot. For the time being, the "desired trajectory" approach seems to be too restricted for general-purpose reactive motor control and planning.

From the viewpoint of statistical learning, Equation (1) constitutes a nonlinear function approximation problem. A typical approach to learning complex nonlinear functions is to compose them out of basis functions of reduced complexity. The same line of thinking generalizes to learning policies: a complicated policy could be learned from the combination of simpler policies, i.e., policy primitives or movement primitives, as for instance:

$$\mathbf{u} = \pi\left(\mathbf{x}, \alpha, t\right) = \sum_{k=1}^{K} \pi_k\left(\mathbf{x}, \alpha_k, t\right) \tag{3}$$

Indeed, related ideas have been suggested in various fields of research, for instance in computational neuroscience as Schema Theory[21], reinforcement learning as macro action or options[38], and mobile robotics as behavior-based or reactive robotics[39]. In particular, the latter approach also emphasized to remove the explicit time dependency of π, such that complicated "clocking" and "reset clock" mechanisms can be avoided, and the combination of policy primitives becomes simplified. Despite the successful application of policy primitives in the mobile robotics domain, so far, it remains a topic of ongoing research [19,20,40] how to generate and combine primitives in a principled and autonomous way, and how such an approach generalizes to complex movement systems, like human arms and legs.

Thus, a key research topic, both in biological and artificial motor control, revolves around the question of movement primitives: what is a good set of primitives, how can they be formalized, how can they interact with perceptual input, how can they be adjusted autonomously, how can they be combined task specifically, and what is the origin of primitives? In order to address the first four of these questions, we suggest to resort to some of the most basic

ideas of dynamic systems theory. A dynamic system can generally be written as a differential equation:

$$\dot{\mathrm{x}} = f\left(\mathbf{x}, \alpha, t\right) \tag{4}$$

which is almost identical to Equation (1), except that the left-hand-side denotes a change-of-state, not a motor command. Such a kinematic formulation is, however, quite suitable for motor control if we conceive of this dynamic system as a kinematic planning policy, whose outputs are subsequently converted to motor commands by an appropriate controller (Figure 3). Planning

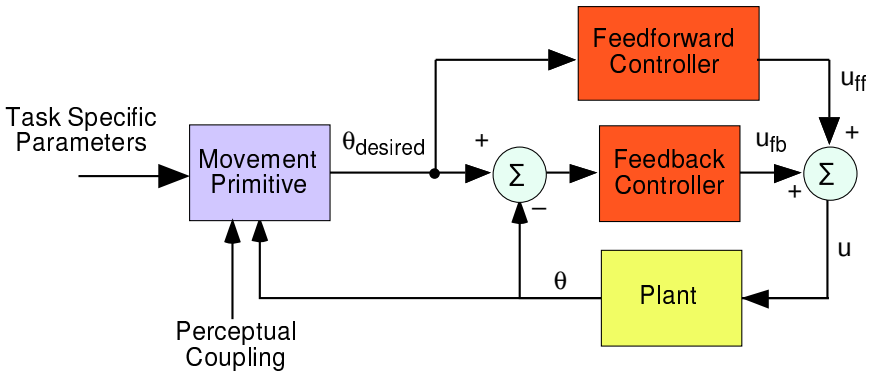

Fig. 3. Sketch of a control diagram with dynamic movement primitives.

in kinematic space is often more suitable for motor control since kinematic plans generalize over a large part of the workspace — nonlinearities due to gravity and inertial forces are taken care off by the controller at the motor execution stage (cf. Figure 2). Kinematic plans can also be theoretically cleanly superimposed to use multiple movement primitives to form more complex behaviors, as indicated in Equation (3). It should be noted, however, that a kinematic representation of movement primitives is not necessarily independent of the dynamic properties of the limb. Proprioceptive feedback can be used to on-line modify the attractor landscape of a DMP in the same way as perceptual information [41–43]. Figure 3 indicates this property with the "perceptual coupling" arrow — the biped locomotion example in Section 4.3 will clarify this issue.

The two most elementary behaviors of a nonlinear dynamic system are point attractive and limit cycle behaviors, paralleled by discrete and rhythmic movement in motor control. The idea of dynamic movement primitives (DMP) is to exploit well-known simple formulations of such attractor equations to code the basic behavioral pattern (i.e., rhythmic or discrete), and to use statistical learning to adjust the attractor landscape of the DMP to the

detailed needs of the task. As will be outlined in the next section, several appealing properties, such as perception-action coupling and reusability of the primitives, can be accomplished in this framework.

3 Dynamic Movement Primitives

3.1 Control with DMPs

We assume that the attractor landscape of a DMP represents the desired kinematic state of a limb, i.e., desired positions, velocities, and accelerations for each joint, or, alternatively, for each coordinate in task space. As shown in Figure 3, kinematic variables are converted to motor commands through an inverse dynamics model and stabilized by low gain feedback control. The example of Figure 3 corresponds to a classical computed torque controller[44], but more advanced learning and/or adaptive controllers can be employed[2]. Thus, the motor execution of DMPs can incorporate any standard control technique that takes as input kinematic trajectory plans.

3.2 Planning with DMPs

In order to accommodate discrete and rhythmic movement plans, two kinds of DMPs are needed: point attractive systems and limit-cycle systems. The key question of DMPs is how to formalize nonlinear dynamic equations such that they can be flexibly adjusted to represent arbitrarily complex motor behaviors without the need for manual parameter tuning and the danger of instability of the equations. We will sketch our approach in the example of a discrete dynamic system for reaching movements — an analogous development holds for rhythmic systems.

Assume we have a basic point attractive system, for instance, instantiated by the second order dynamics

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z), \quad \tau \dot{y} = z + f \tag{5}$$

where g is a known goal state, α_z and β_z are time constants, τ is a temporal scaling factor (see below) and y, $\dot{y}$ correspond to the desired position and velocity generated by the equations, interpreted as a movement plan. For appropriate parameter settings and $f = 0$, these equations form a globally stable linear dynamic system with g as a unique point attractor. Could we find a nonlinear function f in Equation (5) to change the rather trivial exponential convergence of y to allow more complex trajectories on the way to the goal? As such a change of Equation (5) enters the domain of nonlinear dynamics, an arbitrary complexity of the resulting equations can be expected. To the best of our knowledge, this problem has prevented research from employing generic learning in nonlinear dynamic systems so far. However, the introduction of an additional canonical dynamic system (x, v)

$$\tau \dot{v} = \alpha_v(\beta_v(g - y) - v), \quad \tau \dot{x} = v \tag{6}$$

and the nonlinear function f

$$f(x,v,g) = \frac{\sum_{i=1}^{N} \psi_i w_i v_i}{\sum_{i=1}^{N} \psi_i}, \text{ where } \psi_i = \exp\left(-h_i \left(\frac{x}{g} - c_i\right)^2\right) \tag{7}$$

alleviates this problem. Equation (6) is a second order dynamic system similar to Equation (5), however, it is linear and not modulated by a nonlinear function, and, thus, its monotonic global convergence to g can be guaranteed with a proper choice of α_v and β_v, e.g., such that Equation (6) is critically damped. Assuming that all initial conditions of the state variables x,v,y,z are zero, the quotient $x/g \in [0,\ 1]$ can serve as a phase variable to anchor the Gaussian basis functions ψ_i (characterized by a center c_i and bandwidth h_i), and v can act as a "gating term" in the nonlinear function (7) such that the influence of this function vanishes at the end of the movement. Assuming boundedness of the weights w_i in Equation (7), it can be shown that the combined system in Equations (5), (6), and (7) asymptotically converges to the unique point attractor g.

It is not the particular instantiation in Equations (5), (6), and (7) what is the most important idea of DMPs, but rather it is design principle what matters. A DMP consists of two sets of differential equations: a *canonical* system

$$\tau \dot{\mathbf{x}} = \mathbf{h}(\mathbf{x}) \tag{8}$$

and a *transformation* system

$$\tau \dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}, f) \tag{9}$$

The canonical system needs to generate two quantities: a phase variable[1] x, and a phase velocity v, i.e., $\mathbf{x} = [\mathbf{x}\ \mathbf{y}]^{\mathbf{T}}$. The phase x is a substitute for time and allows us anchoring our spatially localized basis functions (7). The appealing property of using a phase variable instead of an explicit time representation is that we can manipulate the time evolution of phase, e.g., by additive coupling terms or phase resetting (cf. Section 4.3) — in contrast, time cannot be manipulated easily. The phase velocity v is a multiplicative term in the nonlinearity (7). If v is set to zero, the influence of the nonlinearity vanishes in the transformation system, and the dynamics of the transformation system with $f = 0$ dominate its time evolution. In the design of a DMP, we usually choose a structure for canonical and transformation systems that are analytically easy to understand, such that the stability properties of the

[1] A phase variable, in our notation, monotonically increases (or decreases) its value from movement start to end under unperturbed conditions. For periodic motion, it may reset after one period to its initial value.

DMP can be guaranteed. In the following, we give the equations for the canonical and transformation systems for another formulation of a discrete system, and also rhythmic systems.

A Discrete Acceleration DMP The canonical system and the function approximator are identical to Equation (6) and Equation (7), respectively. The transformation system is:

$$\begin{aligned} \tau \dot{z} &= \alpha_z(\beta_z(r-y)-z)+f \\ \tau \dot{y} &= z \\ \tau \dot{r} &= \alpha_g(g-r) \end{aligned} \tag{10}$$

This set of equations moved the nonlinear function into the differential equation of $\dot{z}$. Thus, $\dot{z}$, z, y can be interpreted as the desired acceleration, velocity, and position, i.e., $\ddot{y}$, $\dot{y}$, y, generated by the DMP. In order to ensure a continuous acceleration profile for $\ddot{y}$, we had to introduce a simple first order filter for the goal state in the $\dot{r}$ equation – if α_z and β_z are chosen for critical damping, i.e., $\beta_z = \alpha_z/4$, then $\alpha_g = \alpha_z/2$ is a suitable time constant for the $\dot{r}$ equation for a three-fold repeated eigenvalue of $\lambda_{1,2,3} = \alpha_z/2$ of the linear system in Equation (10) without f. The advantage of this "acceleration" DMP is that it can easily be used in conjunction with an inverse model controller that requires a continuous desired acceleration signal. As a disadvantage, the complexity of the transformation system had to be increased to 3^{rd} order.

A Phase Oscillator DMP By replacing the point attractor in the canonical system with a limit cycle oscillator, a rhythmic DMP is obtained[29]. Among the simplest limit cycle oscillators is a phase-amplitude representation:

$$\tau \dot{r} = \alpha_r(A-r),\ \tau\dot{\phi} = 1 \tag{11}$$

where r is the amplitude of the oscillator, A the desired amplitude, and ϕ its phase. For this case, Equation (7) is modified to

$$\begin{aligned} f(r,\ \phi) &= \frac{\sum_{i=1}^{N} \psi_i \mathbf{w}_i^T \mathbf{v}}{\sum_{i=1}^{N} \psi_i}, \text{ where } \mathbf{v} = [r\cos\phi,\ r\sin\phi]^T \\ \text{and } \psi_i &= \exp\left(-h_i(\text{mod}(\phi,\ 2\pi) - c_i)^2\right) \end{aligned} \tag{12}$$

The transformation system in Equations (5) remains the same, except that we now identify the goal state g with a setpoint around which the oscillation takes place. Thus, by means of A, τ, and g, we independently can control amplitude, frequency, and setpoint of an oscillation.

A Limit-Cycle Oscillator DMP One slight variant on the rhythmic DMP above is to employ an "energy-based" limit cycle oscillator as the canonical system [45,46]

$$\begin{aligned} \tau \dot{v} &= -\alpha \frac{E - E_0}{E_0} v - k^2 x \\ \tau \dot{x} &= v, \text{ where } E = (v^2 + k^2 x^2) \end{aligned} \tag{13}$$

with nonlinear function

$$\begin{aligned} f(x,v) &= \frac{\sum_{i=1}^{N} \psi_i \mathbf{w}_i^T \mathbf{v}}{\sum_{i=1}^{N} \psi_i}, \text{ where } \mathbf{v} = \begin{bmatrix} v & \sqrt{E_0} \end{bmatrix}^T \\ \text{and } \psi_i &= \exp\left(-h_i(\arctan 2(v,x) - c_i)^2\right) \end{aligned} \tag{14}$$

E_0 is interpreted as the desired energy level of the oscillator; it determines the amplitude of the oscillation, while k determines its frequency. The difference between the two oscillator formulations comes to bear when perturbation and coupling effects need to be considered. As will be demonstrated in a later section, the phase oscillator formulation is particularly useful for modeling effects like phase resetting and phase coupling, while the energy-based oscillator is more useful when effects like "synaptic coupling" need to be incorporated[47].

3.3 Invariance Properties of DMPs

In all the different DMP variants above, the weights $\mathbf{w}_i$ determine the particular shape of the trajectories realized by the DMP, the parameter τ the speed of a movement, and some other parameters like g, E_0, or A the amplitude of the movement. In order to exploit the property that DMPs code kinematic control policies, i.e., the plans can theoretically be re-used in many parts of the workspace, it is desirable that DMPs retain their qualitative behavior if translated and if scaled in space or time. From a dynamic systems point of view, we wish that the attractor landscape of a DMP does not change qualitatively after scaling, a topic addressed in the framework of "topological equivalence"[48]. Formally, if dynamic system one obeys $\dot{\mathbf{x}} = f(\mathbf{x})$, and system two yields $\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y})$, then the existence of an orientation preserving homeomorphism h: $[\mathbf{x}, \dot{\mathbf{x}}] \xrightarrow{h} [\mathbf{y}, \dot{\mathbf{y}}]$ and $[\mathbf{x}, \dot{\mathbf{x}}] \xleftarrow{h^{-1}} [\mathbf{y}, \dot{\mathbf{y}}]$ proves topological equivalence. It can easily be verified, that if a new DMP is created by multiplying τ, g, E_0, or A in any of the DMPs above by a factor c, a simple multiplication of all state variables and change-of-state variables by a factor c or $\sqrt{c}$ constitutes the required homeomorphism to proof topological equivalence.

Thus, if designed correctly, DMPs can be re-used in new situation similar to the one for which they were built. For instance, a tennis forehand DMP could be used to create a small amplitude swing, or a large amplitude swing, without changing the basic swing pattern that the tennis coach so carefully taught to the student.

3.4 Imitation Learning with DMPs

An important issue is how to learn the weights $\mathbf{w}_i$ in the nonlinear function f that characterizes the spatiotemporal path of a DMP. Given that f is a normalized basis function representation with linear parameterization[49], a variety of learning algorithms exist to find $\mathbf{w}_i$. Let us assume we are given a sample trajectory $y_{demo}(t)$, $\dot{y}_{demo}(t)$, $\ddot{y}_{demo}(t)$ with duration T, e.g., as typical in imitation learning[9]. Based on this information, a supervised learning problem results with the following target for f:

- For the transformation system in Equation (5), using $g = y_{demo}(T)$:

$$\begin{aligned} f_{target} &= \tau \dot{y}_{demo} - z_{demo} \text{ where} \\ \tau \dot{z}_{demo} &= \alpha_z(\beta_z(g - y_{demo}) - z_{demo}) \end{aligned} \tag{15}$$

- For the transformation system in Equation (10)

$$f_{target} = \tau \ddot{y}_{demo} - \alpha_z(\beta_z(r - y_{demo}) - \dot{y}_{demo}) \tag{16}$$

In order to obtain a matching input for f_{target}, the canonical system needs to be integrated. For this purpose, in Equation (6), the initial state of the canonical system is set to $v = 0$, $x = y_{demo}(0)$ before integration. An analogous procedure is performed for the rhythmic DMPs. The time constant τ is chosen such that the DMP with $f = 0$ achieves 95% convergence at $t = T$. With this procedure, a clean supervised learning problem is obtained over the time course of the movement to be approximated with training samples $(\mathbf{v}, f_{target})$ (cf. Equations (7),(12),(14)).

For solving the function approximation problem, we chose a nonparametric regression technique from locally weighted learning (LWPR)[50] as it allows us to determine the necessary number of basis functions N, their centers c_i, and bandwidth h_i automatically. In essence, for every basis function ψ_i, LWPR performs a locally weighted regression of the training data to obtain an approximation of the tangent of the function to be approximated within the scope of each basis function. Predictions for a query point are generated by a ψ_i-weighted average of the predictions of all local models. Given that the parameters $\mathbf{w}_i$ learned by LWPR are independent of the number of basis functions, they can be used robustly for categorization of DMPs (see Section 3.8).

In summary, by anchoring a linear learning system with nonlinear basis functions in the *phase space* of a *canonical dynamic system with guaranteed*

attractor properties, we are able to learn complex attractor landscapes of nonlinear differential equations without endangering the asymptotic convergence to the goal state. Both discrete and rhythmic movements can thus be coded in the DMPs with almost arbitrarily complex smooth trajectory profiles. This strategy opens a large range of possibilities to create movement primitives, e.g., for reaching, grasping, object manipulations, and locomotion.

3.5 Reinforcement Learning of DMPs

While imitation learning provides an excellent means to start a movement skill at a high performance level, many movement tasks require trial-and-error refinement until a satisfying skill level is accomplished. From the viewpoint of DMPs, we need non-supervised learning methods to further improve the weights $\mathbf{w}=\{\mathbf{w}_i\}$, guided by a general reward or optimization criterion.

Optimization theory, dynamic programming and reinforcement learning [5,32,51] offer a general set of tools for learning $\mathbf{w}$ from trial and error, also called "roll-outs" in reinforcement learning. We assume that an arbitrary optimization criterion J governs our learning process, such that it is not possible to obtain analytical gradients $dJ/d\mathbf{w}$. Thus, the gradient needs to be estimated from empirical data, i.e., by trying a certain instantiation of the parameters and monitoring its performance. Levenberg-Marquardt and Gauss-Newton algorithms are a possible choice for this task[51] – however, both algorithm often dare large jumps in parameter space under the assumption that an aggressive exploration of parameters is permissible. For motor learning on a physical robot, we require a safer and smoother learning system, at the cost of slightly slower convergence. For this purpose, we developed a novel reinforcement learning algorithm, the Natural-Actor Critic (NAC)[52]. The NAC is a stochastic gradient method, i.e., it injects noise in the control policy in order to provided the necessary exploration for learning. We will illustrate the NAC algorithm in the context of the acceleration DMP in Section 3.2.

The DMP in Equation (10) can be interpreted as creating an acceleration command $\bar{\ddot{y}} = \dot{z}$ in the top equation of (10). In the NAC, this acceleration command is treated as the mean of a Gaussian control policy

$$\pi(\ddot{y}|x,v,z,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}\left(\ddot{y} - \bar{\ddot{y}}\right)^2\right) \tag{17}$$

with variance σ^2. Given a reward $r(\ddot{y}, x, v, y, z)$ at every time step of the movement, the goal of learning is to optimize the expected accumulated reward

$$J(\mathbf{w}) = E\left\{\sum_{t=0}^{T} r_t\right\} \tag{18}$$

where the expectation is take with respect to all trajectories starting at the same start state and following the stochastic policy above. As developed in

detail in [52], the *natural* gradient of $dJ/d\mathbf{w}_i$ can be estimated by a linear regression procedure:

Define for one roll-out$r$:

$$R_r = \sum_{t=0}^{T} r_{t,r} \text{ and}$$

$$\phi_r = \sum_{t=0}^{T} \begin{bmatrix} \partial \log \pi(\ddot{y}_t|, x_t, v_t, y_t, z_t)/\partial \mathbf{w} \\ 1 \end{bmatrix}_r \tag{19}$$

After multiple roll-outs, compute$\partial J/d\mathbf{w}$:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \vdots \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \end{bmatrix}, \ \begin{bmatrix} \partial J/d\mathbf{w} \\ c \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \mathbf{R}$$

where c is the regression parameter corresponding to the constant offset in the regression. The *natural* gradient is a special version of the regular gradient that takes into account the Riemannian structure of the space in which the optimization surface lies[53]; as demonstrated in [**?**], it allows a much more efficient form of gradient descent. The above algorithm can also be formulated as a recursive estimation method using recursive least squares [54], and the variance σ^2 of the stochastic policy can be included in the parameters to be optimized. By updating $\mathbf{w}$ with gradient ascent (or descent) according to the natural gradient estimate, fast convergence to a locally optimal parameterization can be accomplished.

3.6 Multiple Degree-of-Freedom DMPs

So far, our treatment of DMPs focused on single degree-of-freedom (DOF) systems. An extension to multiple DOFs is rather straightforward. The simplest form employs a separate DMP for every DOF. Alternatively, all DMPs could share the same canonical system, but have separate transformation systems, as realized in [46]. In this case, every DOF learns its own function f. By sharing the same canonical system, very complex phase relationships between individual DOFs can be realized and stabilized, for instance, as needed for biped locomotion in the examples in Section 4.

3.7 Superposition of DMPs

Given that DMPs create kinematic control polices, a superposition of DMPs to generate behaviors that are more complex is possible. For instance, a discrete DMP could be employed to shift the setpoint of a rhythmic DMP, thus generating a point-to-point movement with a superimposed periodic pattern. For example, with this strategy is possible to bounce a ball on a racket by producing an oscillatory up-and-down movement in joint space of the arm, and use the discrete system to make sure the oscillatory movement

remains under the ball such that the task can be accomplished[41,55]. Other forms of superposition are conceivable, and future work will evaluate the most promising strategies.

3.8 Movement Recognition with DMPs

The invariance properties described in Section 3.3 render the parameters $\mathbf{w}$ of a DMP insenstive towards movement translation and spatial and temporal scaling. Thus, the $\mathbf{w}$ vector can serve as a classifier for DMPs, e.g., by using nearest neighbor classification or more advanced classification techniques[49]. In [30], we demonstrated how DMPs can be used for character recognition of the Palm Pilot graffiti alphabet.

4 Evaluations

The following sections give some examples of the abilities of DMPs in the context of humanoid robotics. We implemented our DMP system on a 30 DOF Sarcos Humanoid robot (Figure 1). Desired position, velocity, and acceleration information was derived from the states of the DMPs to realize a compute-torque controller (Figure 3). All necessary computations run in real-time at 420Hz on a multiple processor VME bus operated by VxWorks.

4.1 Imitation Learning

In [30], we demonstrated how a complex tennis forehand and tennis backhand swing can be learned from a human teacher, whose movements were captured at the joint level with an exoskeleton. Figure 4 illustrates imitation learning for a rhythmic trajectory using the phase oscillator DMP from Section 3.2. The images in the top of Figure 4 show four frames of the motion capture of a figure-8 pattern and its repetition on the humanoid robot after supervised learning of the trajectory as described in Section 3.4. The bottom-left plots demonstrate the motion captured and fitted trajectory of a bimanual drumming pattern, using 6 DOFs per arm. All DMPs referred to the same canonical system (cf. Section 3.6). Note that very complex phase relationships between the individual DOFs can be realized. For one joint angle, the right elbow joint (R_EB), the bottom-right plot exemplifies the effect of various changes of parameter settings of the DMP (cf. figure caption in Figure 4). Here it is noteworthy how quickly the pattern converges to the new limit cycle attractor, and that parameter changes do not change the movement pattern qualitatively, as predicted from the analysis of Section 3.3. The nonlinear function of each DMP employed 15 basis functions.

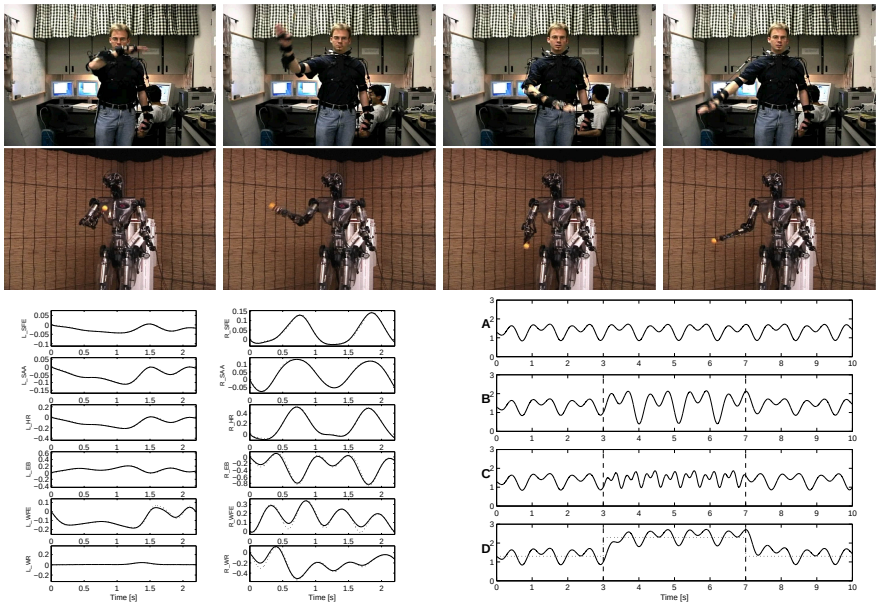

Fig. 4. *Top*: Humanoid robot learning a figure-8 movement from a human demonstration. *Left*: Recorded drumming movement performed with both arms (6 DOFs per arm). The dotted lines and continuous lines correspond to one period of the demonstrated and learned trajectories, respectively – due to rather precise overlap, they are hardly distinguishable. *Right*: Modification of the learned rhythmic pattern (flexion/extension of the right elbow, R_EB). A: trajectory learned by the rhythmic DMP, B: temporary modification with $A \leftarrow 2A$, C: temporary modification with $\tau \leftarrow \tau/2$, D: temporary modification with $g \leftarrow g+1$ (dotted line). Modified parameters were applied between t=3s and t=7s.

4.2 Reinforcement Learning

In a preliminary application of the reinforcement learning method of Section 3.5, we optimized the weights of the acceleration DMP of Section 3.2 to create smooth joint-level trajectories in the spirit of 5^{th} order polynomials [56,57]. The reward per trajectory was

$$R = 1000\left((y(T)-g)^2 + \dot{y}^2(T)\right) + \sum_{t=0}^{T} \ddot{y}^2(t) \tag{20}$$

Weights of each DMP were initialized to zero. Each movement started at $y = 0$ and moved within 500ms to $g = 1$. Figure 5 illustrates the convergence of the Natural Actor Critic algorithm in comparison to a non-natural gradient method, *Episodic Reinforce*[58]. The NAC algorithm converges smoothly with about one order of magnitude faster performance than *Episodic Reinforce*. Good smooth bell-shaped velocity profiles of the DMP are reached after about

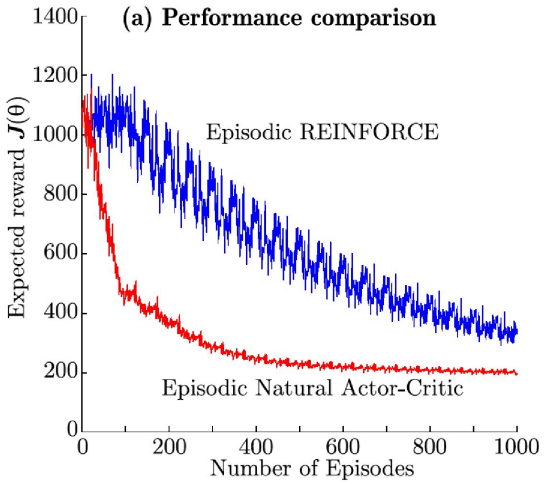

Fig. 5. Convergence of Natural Actor Critic reinforcement learning for learning the weights of a DMP.

150-200 trials, which is comparable to human learning in related tasks[59]. This initial evaluation demonstrates the efficiency of the NAC algorithm for optimizing DMPs, although more thorough evalutions are needed for various reward criteria and also multi-DOF tasks.

4.3 Learning Resonance Tuning in Biped Locomotion

As a last evaluation of DMPs, we applied the phase oscillator DMP to simulated biped locomotion[60] y. Figure 6 shows the setup of the planar biped. Motion capture date from human locomotion was employed to learn an initial trajectory pattern, which, after some modest tuning of the speed and amplitude parameters of the DMP, achieved stable locomotion. Consider the following update law for the phase and frequency of the canonical system of the DMP at the moment of heel-strike:

$$\begin{aligned} \dot{\phi} &= \hat{\omega}^n + \delta(t - t_{heel-strike})(\phi^{robot}_{heel-strike} - \phi) \\ \hat{\omega}^{n+1} &= \hat{\omega}^n + K(\omega^n_{measured} - \hat{\omega}^n) \end{aligned} \tag{21}$$

where δ is the Dirac delta function, $n$is the number of steps, and $\phi^{robot}_{heel-strike}$ is the phase of the mechanical oscillator (robot) at heel strike defined as $\phi^{robot}_{heel-strike} = 0$ at the heel strike of the leg with the corresponding oscillator, and $\phi^{robot}_{heel_strike} = \pi$ at the heel strike of the other leg. $\omega^n_{measured}$ is the measured frequency of locomotion defined by $\omega^n_{measured} = \pi/T_n$, where T_n is the time for one step of locomotion (half period with respect to the oscillator). This equation introduces phase resetting of the DMP at heel-strike as

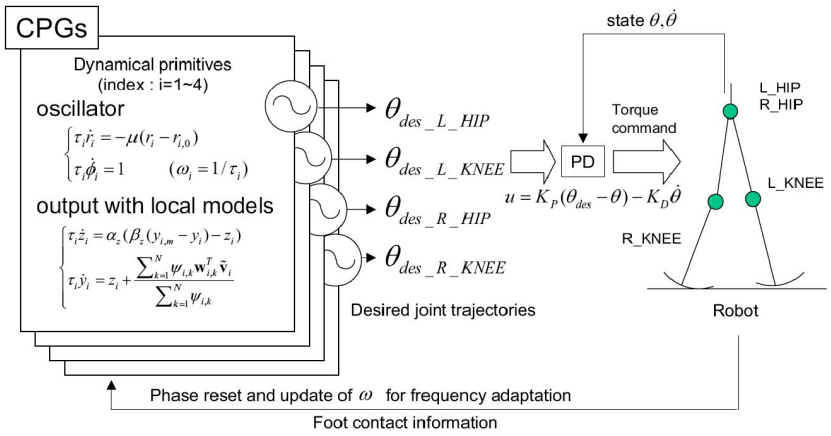

Fig. 6. Four rhythmic DMPs drive the four actuated joints of a planar biped simulation, consisting of two hip and two knee joints. A PD controller is used to track the trajectories generated by the DMPs.

long as the natural frequency of the robot does not correspond to the natural frequency of the canonical system of the DMP; more details can be found in [60]. Figure 7 depicts the time course of adaptation of the movement frequency of the DMPs for the right leg DOFs due to the update law above. The frequency gradually increases until it reaches approximately resonance frequency of the simulated robot legs. This simulation provides thus a nice example how imitation learning can initially be used to start a movement skill, and self-improvement can optimize the pattern for the particular inertial and geometric structure of the robot.

5 Conclusion

This paper described research towards generating flexible movement primitives out of nonlinear dynamic attractor systems. We focused on motivating the design principle of appropriate dynamic systems such that discrete and rhythmic movements could be learned for high-dimensional movement systems. In particular, we emphasized methods of imitation learning and reinforcement learning for acquiring motor skills with movement primitives. We also described some implementations of our methods of dynamic movement primitives on a complex anthropomorphic robot, demonstrating imitation learning of complex rhythmic movement, re-using of learn skills in related situations, and resonance tuning for biped locomotion. We believe that the framework of dynamic movement primitives has a tremendous potential for understanding autonomous generation of complex motor behaviors in humans

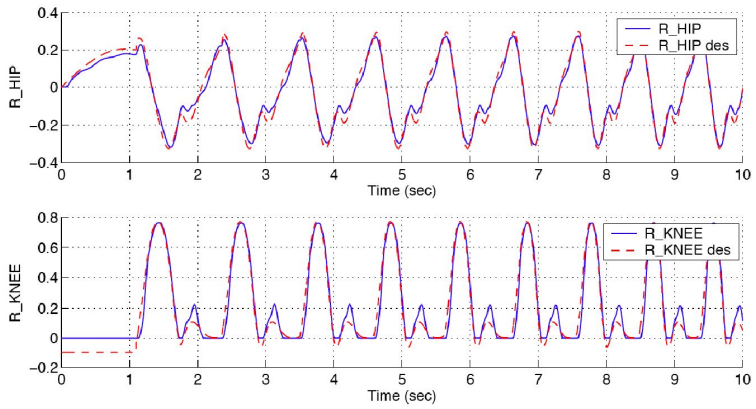

Fig. 7. Time course of trajectories of right leg DOFs using the phase resetting adaptation law. The movement frequency slow decreases until resonance tuning is accomplished.

and humanoids. Our future work will pursue a combination of robotic, theoretical, and biological research addressing how a library of motor primitives can be created, maintained, adapted to new situations, and sequenced and superimposed for the creation of complex perceptuomotor behaviors.

6 Acknowledgments

This work was made possible by awards #9710312/#0010312 and #0082995 of the National Science Foundation, award AC#98-516 by NASA, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Agency, and the ATR Human Information Processing Research Laboratories.

References

1. P. Menzel and F. D'Alusio (2000) Robosapiens: Evolution of a new species, Cambridge, MA: MIT Press
2. J. Nakanishi, J. A. Farrell, and S. Schaal (submitted) Composite adaptive control with locally weighted statistcal learning, International Journal of Robotics Research
3. S. M. LaValle (2003) Planning algorithms
4. J.-C. Latombe (1991) Robot motion planning, Boston: Kluwer Academic Publishers
5. R. S. Sutton and A. G. Barto (1998) Reinforcement learning : An introduction, Cambridge: MIT Press

6. S. Thrun (2000), Probabilistic algorithms in robotics, AI Magazine, vol. 21, pp. 93-109
7. O. Khatib, K. Yokio, O. Brock, K. Chang, and A. Casal, (2001) Robots in human environments, Archives of Control Sciences, Special Issue on Granular Computing, vol. 11, pp. 123-128
8. C. G. Atkeson, J. Hale, M. Kawato, S. Kotosaka, F. Pollick, M. Riley, S. Schaal, S. Shibata, G. Tevatia, and A. Ude (2000) Using Humanoid Robots to Study Human Behaviour, IEEE Intelligent Systems, vol. 15, pp. 46-56
9. S. Schaal (1999) Is imitation learning the route to humanoid robots?, Trends in Cognitive Sciences, vol. 3, pp. 233-242
10. M. Mataric (2000) Getting humanoids to move and imitate, IEEE Intelligent Systems, vol. 15, pp.18-24
11. K. Dautenhahn and C. L. Nehaniv (2002) Imitation in animals and artifacts, Cambridge, MA: MIT Press
12. S. Schaal, A. Ijspeert, and A. Billard (2003) Computational approaches to motor learning by imitation, Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences, vol. 358, pp. 537-547
13. J. Rittscher, A. Blake, A. Hoogs, and G. Stein (2003) Mathematical modelling of animate and intentional motion, Philos Trans R Soc Lond B Biol Sci, vol. 358, pp. 475-90
14. T. Shibata, S. Vijayakumar, J. Conradt, and S. Schaal (2001) Biomimetic oculomotor control, Adaptive Behavior, vol. 9, pp. 189-207
15. L. Itti and C. Koch (2001) Computational modelling of visual attention, Nat Rev Neurosci, vol. 2, pp. 194-203
16. J. Triesch and C. von der Malsburg (2001) Democratic integration: self-organized integration of adaptive cues, Neural Comput, vol. 13, pp. 2049-74
17. S. Hirose (2002) Two decades of locomotion study: Retrospection and prospect for the future, Journal of the Robotics Society of Japan, vol. 20, pp. 1-6
18. J. Lee, J. Chai, A. Reitsma, J. K. Hodgins, and N. S. Pollard (2002) Interactive control of avatars animated with human motion data, presented at Proceedings of SIGGRAPH
19. R. R. Burridge, A. A. Rizzi, and D. E. Koditschek (1999) Sequential composition of dynamically dexterous robot behaviors, International Journal of Robotics Research, vol. 18, pp. 534-555
20. T. Inamura, I. Toshima, and Y. Nakamura (2002) Acquisition and embodiment of motion elements in closed mimesis loop, presented at International Conference on Robotics and Automation (ICRA2002), Washinton, May 11-15
21. M. A. Arbib (1981) Perceptual structures and distributed motor control, in Handbook of Physiology, Section 2: The Nervous System Vol. II, Motor Control, Part 1, V. B. Brooks, Ed.: Bethesda, MD: American Physiological Society, 1981, pp. 1449-1480
22. P. Viviani (1986) Do units of motor action really exist?, in Experimental Brain Research Series 15. Berlin: Springer, pp. 828-845.
23. M. Mataric (1988) Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior, Trends in Cognitive Sciences, vol. 2, pp. 82-86
24. D. Sternad and D. Schaal (1999) Segmentation of endpoint trajectories does not imply segmented control, Experimental Brain Research, vol. 124, pp. 118-136
25. H. Miyamoto and M. Kawato (1998) A tennis serve and upswing learning robot based on bi-directional theory, Neural Networks, vol. 11, pp. 1331-1344

26. S. Schaal (2002) Learning robot control, in The handbook of brain theory and neural networks, 2nd Edition, M. A. Arbib, Ed., 2 ed. Cambridge, MA: MIT Press, pp. 983-987
27. S. Schaal Arm and hand movement control, in The handbook of brain theory and neural networks, 2nd Edition, M. A. Arbib, Ed., 2 ed. Cambridge, MA: MIT Press, 2002, pp. 110-113.
28. A. Ijspeert, J. Nakanishi, and S. Schaal, Trajectory formation for imitation with nonlinear dynamical systems, presented at IEEE International Conference on Intelligent Robots and Systems, 2001.
29. A. 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.: Cambridge, MA: MIT Press, 2003.
30. J. A. Ijspeert, J. Nakanishi, and S. Schaal, Movement imitation with nonlinear dynamical systems in humanoid robots, presented at International Conference on Robotics and Automation, Washinton, May 11-15 2002.
31. R. Bellman, Dynamic programming. Princeton, N.J.: Princeton University Press, 1957.
32. P. Dyer and S. R. McReynolds, The computation and theory of optimal control. New York: Academic Press, 1970.
33. G. Tesauro, Temporal difference learning of backgammon strategy, in Proceedings of the Ninth International Workshop Machine, D. Sleeman and P. Edwards, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 9-18.
34. D. P. Bertsekas and J. N. Tsitsiklis, Neuro-dynamic Programming. Bellmont, MA: Athena Scientific, 1996.
35. J. M. Hollerbach, Dynamic scaling of manipulator trajectories, Transactions of the ASME, vol. 106, pp. 139-156, 1984.
36. S. Kawamura and N. Fukao, Interpolation for input torque patterns obtained through learning control, presented at International Conference on Automation, Robotics and Computer Vision, Singapore, Nov., 1994.
37. K. Yamane and Y. Nakamura, Dynamics filter – conopet and implementation of on-line motion generator for human figures, presented at International Conference on Robotics and Automation, San Francisco, April 2000.
38. R. S. Sutton, D. Precup, and S. Singh, Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning, Artificial Intelligence, vol. 112, pp. 181-211, 1999.
39. R. A. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, vol. 2, pp. 14-23, 1986.
40. W. Lohmiller and J. J. E. Slotine, On contraction analysis for nonlinear systems, Automatica, vol. 6, 1998.
41. A. A. Rizzi and D. E. Koditschek, Further progress in robot juggling: Solvable mirror laws, presented at IEEE International Conference on Robotics and Automation, San Diego, CA, 1994.
42. S. Schaal and D. Sternad, Programmable pattern generators, presented at 3rd International Conference on Computational Intelligence in Neuroscience, Research Triangle Park, NC, 1998.
43. M. Williamson, Neural control of rhythmic arm movements, Neural Networks, vol. 11, pp. 1379-1394, 1998.
44. J. J. Craig, Introduction to robotics. Reading, MA: Addison-Wesley, 1986.

45. J. Nakanishi, T. Fukuda, and D. E. Koditschek, A brachiating robot controller, IEEE Transactions on Robotics and Automation, vol. 16, pp. 109-123, 2000.
46. J. A. Ijspeert, J. Nakanishi, and S. Schaal, Learning rhythmic movements by demonstration using nonlinear oscillators, presented at IEEE International Conference on Intelligent Robots and Systems, Lausanne, Sept.30-Oct.4, 2002.
47. D. Sternad, E. L. Amazeen, and M. T. Turvey, Diffusive, synaptic, and synergetic coupling: An evaluation through inphase and antiphase rhythmic movements, Journal of Motor Behavior, vol. 28, pp. 255-269, 1996.
48. E. A. Jackson, Perspectives of nonlinear dynamics, Vol.1 New York: Cambridge University Press, 1989.
49. C. M. Bishop, Neural networks for pattern recognition. New York: Oxford University Press, 1995.
50. S. Vijayakumar and S. Schaal, Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional spaces, presented at Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, 2000.
51. W. P. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, Numerical recipes in C – The art of scientific computing. Cambridge, MA: Press Syndiacate University of Cambridge, 1989.
52. J. Peters, S. Vijayakumar, and S. Schaal, Reinforcement learning for humanoid robotics, presented at Humanoids2003, Third IEEE-RAS International Conference on Humanoid Robots, Karlsruhe, Germany, Sept.29-30, 2003.
53. S. Amari, Natural gradient learning for over- and under-complete bases In ICA, Neural Comput, vol. 11, pp. 1875-83, 1999.
54. L. Ljung and T. Sderstrm, Theory and practice of recursive identification: Cambridge MIT Press, 1986.
55. S. Schaal, D. Sternad, and C. G. Atkeson, One-handed juggling: A dynamical approach to a rhythmic movement task, Journal of Motor Behavior, vol. 28, pp. 165-183, 1996.
56. L. Sciavicco and B. Siciliano, Modeling and control of robot manipulators. New York: MacGraw-Hill, 1996.
57. R. B. Stein, M. N. Ogusztreli, and C. Capaday, What is optimized in muscular movements?, in Human Muscle Power, N. L. Jones, N. McCartney, and A. J. McComas, Eds. Champaign, Illinois: Human Kinetics Publisher, 1986, pp. 131-150.
58. R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning, vol. 8, pp. 229-256, 1992.
59. R. Shadmehr and F. A. Mussa-Ivaldi, Adaptive representation of dynamics during learning of a motor task, Journal of Neuroscience, vol. 14, pp. 3208-3224, 1994.
60. J. Nakanishi, J. Morimoto, G. Endo, S. Schaal, and M. Kawato, Learning from demonstration and adaptation of biped locomotion with dynamical movement primitives, presented at IEEE International Conference on Intelligent Robots and Systems, Las Vegas, NV, Oct. 27-31, 2003.