Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots

Auke Jan Ijspeert & Jun Nakanishi & Stefan Schaal
Computational Learning and Motor Control Laboratory
University of Southern California, Los Angeles, CA 90089-2520, USA
ATR Human Information Systems Laboratories, Kyoto 619-0288, Japan
Email: ijspeert@usc.edu, nakanisi@rubens.usc.edu, sschaal@usc.edu

Abstract

This article presents a new approach to movement planning, on-line trajectory modification, and imitation learning by representing movement plans based on a set of nonlinear differential equations with well-defined attractor dynamics. In contrast to non-autonomous movement representations like splines, the resultant movement plan remains an autonomous set of nonlinear differential equations that forms a control policy (CP) which is robust to strong external perturbations and that can be modified on-line by additional perceptual variables. The attractor landscape of the control policy can be learned rapidly with a locally weighted regression technique with guaranteed convergence of the learning algorithm and convergence to the movement target. This property makes the system suitable for movement imitation and also for classifying demonstrated movement according to the parameters of the learning system.

We evaluate the system with a humanoid robot simulation and an actual humanoid robot. Experiments are presented for the imitation of three types of movements: reaching movements with one arm, drawing movements of 2-D patterns, and tennis swings. Our results demonstrate (a) that multi-joint human movements can be encoded successfully by the CPs, (b) that a learned movement policy can readily be reused to produce robust trajectories towards different targets, (c) that a policy fitted for one particular target provides a good predictor of human reaching movements towards neighboring targets, and (d) that the parameter space which encodes a policy is suitable for measuring to which extent two trajectories are qualitatively similar.

1 Introduction

This article is a continuation to our quest for robust movement encoding systems to be used with real and simulated humanoid robots. In [1, 2], we identified five desirable properties that such systems should present: 1) the ease of representing and learning a desired trajectory, 2) compactness of the representation, 3) robustness against perturbations and changes in a dynamic environment, 4) ease of re-use for related tasks and easy modification for new tasks, and 5) ease of categorization for movement recognition.

Our approach to fullfil these properties is to encode desired trajectories, or more precisely complete control policies (CPs), in terms of an adjustable pattern generator built from simple nonlinear autonomous dynamical systems. The desired trajectory becomes encoded into the attractor landscape of the dynamical system as a trajectory from an initial state to an end state, in a way which does not require time-indexing and which is robust against perturbations. Locally weighted regression is used to learn the trajectory in a single shot. In contrast to other approaches in the literature (see a review in [1]), the dynamical systems encode *desired* trajectories, not motor commands, such that an additional movement execution stage by means of a standard tracking controller (e.g., inverse dynamics controller in our work) is needed. This strategy, however, does not prevent us from modifying the trajectory plans on-line through external variables, as will be demonstrated later.

We apply the CPs to a task involving the imitation of human movements by a humanoid simulation and a humanoid robot. This experiment is part of a project in rehabilitation robotics —the Virtual Trainer project — which aims at using humanoid rigid body simulations and humanoid robots for supervising rehabilitation exercises in stroke-patients. This article presents three experiments demonstrating how trajectories recorded from human subjects can be reproduced using the CPs. The purpose of the experiments is three-fold. First, we quantify how well these different trajectories can be fitted with our CPs. Second, we estimate to which extent a CP, which is fitted to a particular reaching movement, is a good predictor of the movements performed by the human subject towards neighboring targets. And third, we compare the parameters of the dynamical system fitted to different trajectories in order to evaluate how similar the trajectories are in parameter space. This last point aims at demonstrating that the CPs are not only useful for encoding trajectories, but also for classifying them.

2 Dynamical Systems As Control Policies

For the purpose of learning how to imitate human-like movement, we choose to represent movement in kinematic coordinates, e.g., joint angles of a robot or Cartesian coor-

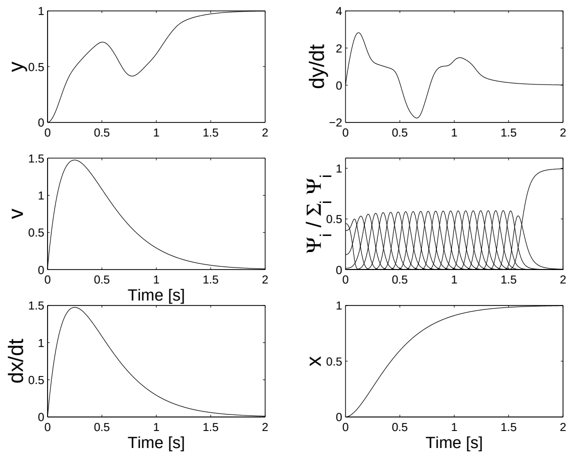

Figure 1: Time evolution of the dynamical systems, with N=25, α_z=8 [1/s], α_v=8 [1/s], β_z=2 [1/s], and β_v=2 [1/s]. The c_i are spread out between c_1=0.0 and c_N=0.99 such as to be equally spaced in time. The same parameters will be used throughout the article. In this particular example, g=1 and the parameters w_i were set by hand to arbitrary (positive and negative) values.

dinates of an endeffector —indeed only kinematic variables are observable in imitation learning. Thus, CPs represent kinematic movement plans, i.e., a change of position as a function of state, and we assume that an appropriate controller exist to convert the kinematic policy into motor commands. A control policy is defined by the following (z, y) dynamics which specify the attractor landscape of the policy for a trajectory y towards a goal g:

$$\dot{z} = \alpha_z(\beta_z(g - y) - z) \quad (1)$$

$$\dot{y} = z + \frac{\sum_{i=1}^{N} \Psi_i w_i}{\sum_{i=1}^{N} \Psi_i} v \quad (2)$$

This is essentially a simple second-order system with the exception that its velocity is modified by a nonlinear term (the second term in equation 2) which depends on internal states. These two internal states, (v, x) have the following second-order linear dynamics

$$\dot{v} = \alpha_v(\beta_v(g - x) - v) \quad (3)$$

$$\dot{x} = v \quad (4)$$

The system is further determined by the positive constants $\alpha_v, \alpha_z, \beta_v$, and β_z, and by a set of N Gaussian kernel functions Ψ_i

$$\Psi_i = \exp\left(-\frac{1}{2\sigma_i^2}(\tilde{x} - c_i)^2\right) \quad (5)$$

where $\tilde{x} = (x - x_0)/(g - x_0)$ and x_0 is the value of x at the beginning of the trajectory. The value x_0 is set each time a new goal is fed into the system, and we assume that $g \neq x_0$, i.e. that the total displacement between the beginning and the end of a movement is never exactly zero. The attractor landscape of the policy can be adjusted by learning the parameters w_i using locally weighted regression [3], see Section 3. Note that this dynamical system has a unique equilibrium point at $(z, y, v, x) = (0, g, 0, g)$.

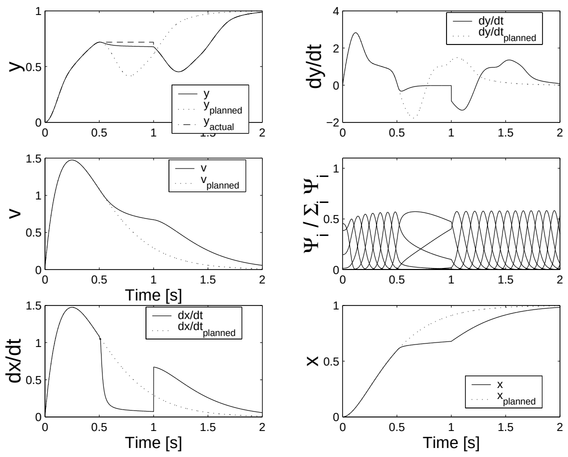

Figure 2: Time evolution of the dynamical systems under a perturbation. The actual position $\tilde{y}$ is frozen between 0.5 and 1.0s. For this example, $\alpha_{py} = 20$ and $\alpha_{px} = 200$ are used in (6) and (7) respectively.

The system is spatially invariant in the sense that a scaling of the goal g does not affect the topology of the attractor landscape. In other words, the position and velocity of a movement will be scaled by a factor c, if a new goal cg is given to the system. This property will be used for goal modulation in the experiment presented in Section 4.2.

Figure 1 demonstrates an exemplary time evolution of the equations. We identify the state y as the desired position that the policy outputs to the controller, and $\dot{y}$ as the desired velocity. The internal states (v, x) are used as follows: x implements a timing signal that is used to localize the Gaussians (5), and v is a scaling signal which ensures that the non-linearity introduced by the Gaussians in Eq. 2 remains transient (v starts and returns to zero at the beginning and end of the discrete movement). v therefore guarantees the unique point attractor of (z, y) at $(0, g)$. As is indicated by the reversal of movement direction in Figure 1, the internal states and the Gaussians allow generating much more complex policies than a policy that has no internal state. As will be demonstrated later, the choice of representing a timing signal in form of autonomous differential equations will allow us to be robust towards external perturbations of the movement–essentially, we can slow down, halt, or even reverse "time" by means of coupling terms in the (v, x) dynamics (see Section 2.2).

2.1 Stability

Stability can be analyzed separately for the (x, v) and the (y, z) dynamics. It is clear that the (x, v) dynamics is globally asymtotically stable with the choice of positive constants. Formally, by following the input-to-state stability analysis in [4], we can conservatively conclude that the equilibrium state of the entire system is asymptotically stable. Intuitively, when $t \to \infty$, we have $\dot{z} = \alpha_z(\beta_z(g-y)-z)$ and $\dot{y} = z$ since $\sum_{i=1}^{N} \Psi_i w_i / \sum_{i=1}^{N} \Psi_i$ is bounded and $v \to 0$ as $t \to \infty$, Thus, it can be seen that $(z, y) \to (0, g)$ as $t \to \infty$. We are currently analyzing the domain of attraction.

2.2 Robustness against Perturbations during Movement Execution

In the current form, the CP would create a desired movement irrespective of the ability of the controller to track the desired movement. When considering applications of our approach to physical systems, e.g., robots and humanoids, interactions with the environment may require an on-line modification of the policy. As an example, we show how a simple feedback coupling term can implement a stop in the time evolution of the policy.

Let $\tilde{y}$ denote the actual position of the movement system. We introduce the error between the planned trajectory y and $\tilde{y}$ to the differential equations (2) and (4) in order to modify the planned trajectory according to external perturbations:

$$\dot{y} = \alpha_y \left(\frac{\sum_{i=1}^{N} \Psi_i w_i}{\sum_{i=1}^{N} \Psi_i} v + z \right) + \alpha_{py}(\tilde{y} - y) \qquad (6)$$

$$\dot{x} = v \left(1 + \alpha_{px}(\tilde{y} - y)^2\right)^{-1} \qquad (7)$$

Figure 2 illustrates the effect of a perturbation where the actual position is artificially fixed during a short period of time. During the perturbation, the time evolution of the states of the policy is gradually halted. The desired position y is modified to remain close to the actual position $\tilde{y}$, and, as soon as the perturbation stops, rapidily resumes performing the (time-delayed) planned trajectory. Note that other ways to cope with perturbations can be designed. Such on-line modifications are one of the most interesting properties of using autonomous differential equations for control policies.

3 Learning From Imitation And Classification

Assume we extracted a desired trajectory y_{demo} from the demonstration of a teacher. Adjusting the CP to embed this trajectory in the attractor landscape is a nonlinear function approximation problem. The demonstrated trajectory is shifted to a zero start position, and the time constants of the dynamical system are scaled such that the time dynamics (v, x) reaches $(0, g)$ at approximately the same time as the goal state is reached in the target trajectory. Given that the goal state is known, it is guaranteed that the policy will converge to the goal; only the time course to the goals needs to be adjusted.

At every moment of time, we have v as input to the learning system, and $u_{des} = \dot{y}_{demo} - z$ as desired output (cf Eq. 2). Locally weighted regression corresponds to finding the parameter w_i which minimizes the locally weighted error criterion $J_i = \sum_t \Psi_i^t (u_{des}^t - u_i^t)^2$, for each local model $u_i^t = w_i v^t$ (i.e. for each kernel function Ψ_i^t), where t is an index corresponding to discrete time steps.[1] Assuming that the goal g is known, this minimum can be computed recursively using locally weighted recursive least squares [3]. Note that locally weighted regression is different from learning with radial basis function networks. One of the significant differences is that in locally weighted learning the parameter w_i for each local model is learned totally independently of all other local models while in learning with RBFs the parameters are learned cooperatively. From a statistical point of view, unlike RBF systems, locally weighted learning methods are robust against overfitting.

[1]In our experiments, the time step size corresponds to the integration step (1ms). The recorded movements are linearly intrapolated to the same step size.

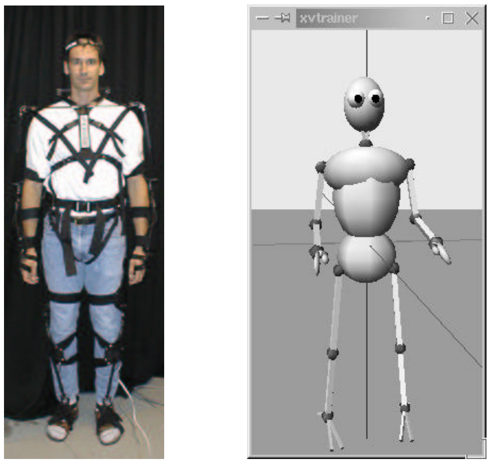

Figure 3: Left: Sensuit for recording joint-angle data. Right: Rigid body simulation.

4 Experimental evaluations

4.1 Humanoid robotics for rehabilitation

One of the motivations to develop the CPs comes from our Virtual Trainer (VT) project. The aim of this project is to supervise rehabilitation exercises in stroke-patients by (a) computerized learning of the exercises from demonstrations by a professional therapist, (b), demonstrating the exercise to the patient with a humanoid simulation, (c) video-based monitoring the patient when performing the exercise, and (d) evaluating the patient's performance, and suggesting and demonstrating corrections with the simulation when necessary.

The essential ingredients of VT are (a) a humanoid robot (either simulated, see Figure 3 right, or real Figure 9), (b) the CPs for imitation, and (c) a movement execution system. The robot is used to demonstrate the exercise to the patient, as well as to provide visual feedback by replicating the movements performed by the patient. The CPs are responsible for determining the kinematics of the desired trajectory, while the movement execution system (an inverse dynamics algorithm) is responsible for the correct production of the trajectory by the dynamic simulation.

As part of the VT project, the requirements for the CPs are the following. First, they must be able to fit the trajectories of the demonstrated movements with high precision. This implies fitting all degrees of freedom (DOFs) (i.e. not only end-points such as the hands), and all points of the trajectories (not only the final positions). Second, they must be robust against perturbations as some exercises require interaction with external objects, e.g., tables or tools. Finally, they must provide a good basis for com-

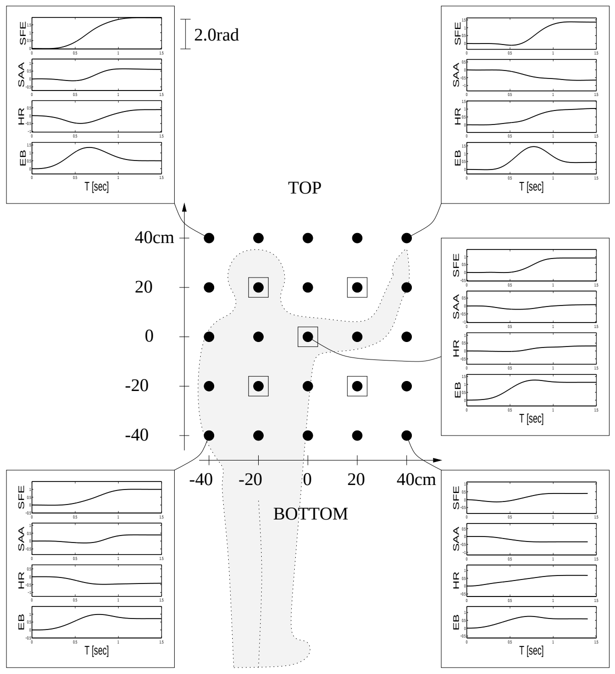

Figure 4: Grid containing the 25 reaching targets with 5 typical reaching trajectories. Each trajectory is represented by the joint angles of four degrees of freedom drawn from top to bottom: shoulder flexion-extension (SFE), shoulder adduction-abduction (SAA), humerus rotation (HR), and elbow flexion-extension (EB). The five points surrounded by a square were used as test sets for the experiment described in Section 4.2.2.

paring movements. One important aspect of VT will be to assess how well the patient performs an exercise, which therefore requires a metric for measuring differences between movements.

4.2 Experiment 1: imitation of reaching movements

The first experiment aimed at systematically testing (1) the fitting of human arm movements, and (2) the modulation of trajectories towards new goals. We recorded trajectories performed by a human subject using a joint-angle recording system, the Sarcos Sensuit (Figure 3 right). The Sensuit directly records the joint angles of 35 DOFS of the human body at 100Hz using hall effect sensors with 12 bit A/D conversion.

We recorded reaching trajectories towards 25 points located on a vertical plane placed 65cm in front of the subject. The points were spaced by 20cm on a 80 by 80cm grid (Figure 4). The middle point was centered on the shoulder, i.e. it corresponded to the (perpendicular) projection of the shoulder onto the plane. The reaching trajectories to each target were repeated three times. Figure 4 shows five typical reaching trajectories in joint space. As could logically be expected, the trajectories present qualitative differences between movements to the left and right, with the shoulder adduction-abduction (SAA) angle increasing and decreasing respectively. The differences in shoulder flexion-extension (SFE) angles and elbow flexion-extension (EB) are essentially quantitative rather than qualitative.

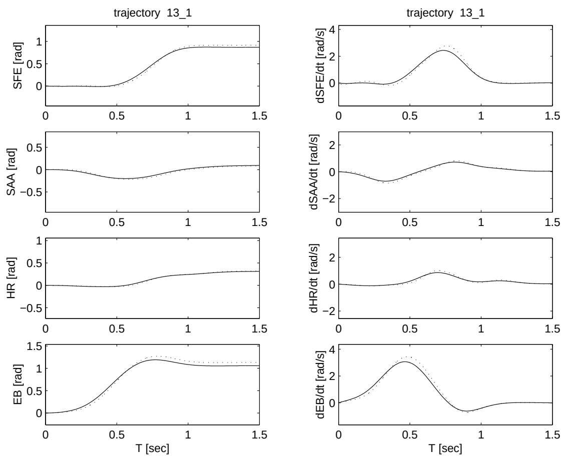

Figure 5: Fitting of a reaching movement towards a target located at <0,0>cm. The recorded and fitted angles of the DOFs are drawn in dotted and continuous lines, respectively.

4.2.1 Fitting each reaching movement

A reaching movement is fitted by the CPs as four independent trajectories, one for each degree of freedom (SFE, SAA, HR and EB). Each of these trajectories is fitted with a system of 25 kernel functions. We fitted the total of 75 reaching movements performed. Figure 5 compares the recorded and fitted trajectories for one particular example. It can be observed that 25 kernel functions are sufficient to fit the human data with little residual error.

4.2.2 Modulation of the goal of the reaching movement

We tested how well the CP fitted to one particular trajectory could be used as a predictor of the trajectories performed by the human subject towards neighboring targets. Five control policies —those fitted to the five targets surrounded by a a square in Figure 4— were tested.

The test is performed as follows. Once a CP is fitted to one particular reaching movement, its parameters are fixed and only the goals g, i.e. the four angles (one per DOF) to which the CP will converge, are modulated. The different goals that we feed into the CP were extracted from the final postures of the human subject when reaching towards different target points. This strategy therefore allows us to directly compare the angular trajectories predicted by the CPs with those performed by the subject. The accuracy of the prediction is measured by computing the square error between the recorded and predicted velocities normalized by the difference between the maximum and minimum velocities of the recorded movement $\sum_{t=0}^{T}(\frac{v_t^{\text{rec}}-v_t^{\text{pred}}}{\max(v_t^{\text{rec}})-\min(v_t^{\text{rec}})})^2$, summed for the four degrees of freedom, and averaged over the three instantiations of each recorded movement (T is the number of time steps).

Figure 6 shows the resulting square error measurements between predictions and recorded movements for the five chosen CPs. Two observations can be made. First, a CP tends to be a better predictor of recorded movements which were made towards a target close to the target for which it was fitted. This is logical, as it simply means that human reaching movements towards close by goals

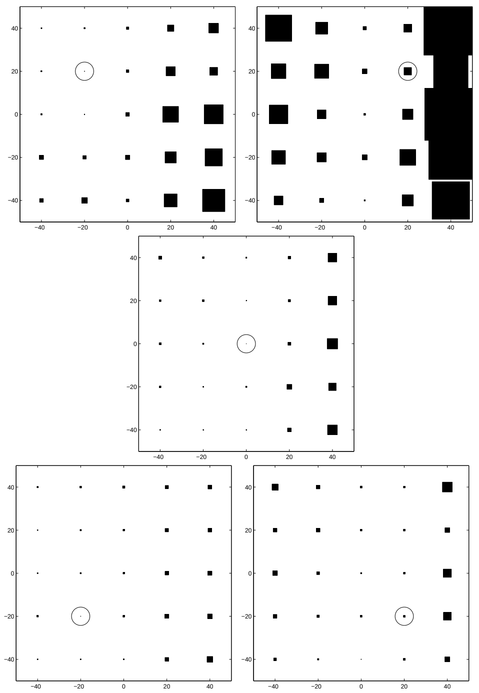

Figure 6: Error between predictions of the CPs and recorded movements. The targets of the recorded movements which were used for fitting each of the five CPs are shown with circles. The widths of the squares are proportional to the squared error measurements as described in the text.

tend to follow similar trajectories.[2] Second, a CP is a significantly better predictor of recorded movements which are made towards targets located above or below the original target, compared to targets located left or right to that target. This is in agreement with our observation that human trajectories in our recordings present qualitative differences between left and right —which therefore require different CPs— and only quantitative differences between top and bottom —which can be represented by a single CP with different goals.

Note that the purpose of our CPs is not to explain human motor control, and there is no reason to expect that a single CP could predict all these human reaching movements. As mentioned above, the movements exhibit qualitative differences in joint space. One would need some cartesian space criterion (e.g straight line of the end point of the arm) and some optimization criterion (e.g. minimum torque change) to explain human motor control. Our purpose is rather to develop a system to be used in humanoid robotics which can accurately encode specific trajectories, replay them robustly, and modulate them to different goals easily. Here we observe that, for the specific trajectories that we recorded from human subjects, a CP fitted towards a particular target has the additional property of being a good predictor for human movements to neighbor targets.

Experiments implementing the trajectories in a real robot can be found in [2]. The movements performed by the robot appear strinkingly human-like. Movies of the recorded trajectories and those performed by the robot can be found at http://www-clmc.usc.edu/~ijspeert/humanoid.html.

[2]Because of the variability of human movements, the prediction error for the recorded trajectories towards the same targets as those used for the fitting are not zero, as the two other instantiations might vary slightly from the trajectory used for the fitting.

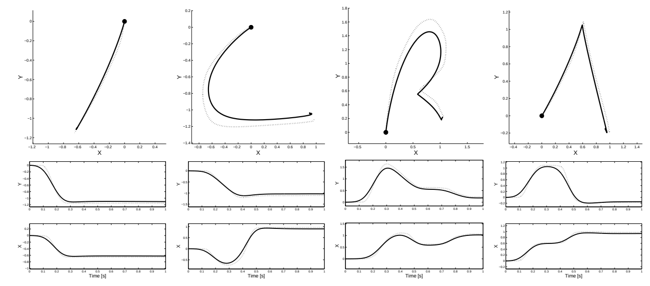

Figure 7: Examples of two-dimensional trajectories fitted by the CPs. The demonstrated and fitted trajectories are shown with dotted and continuous lines, respectively.

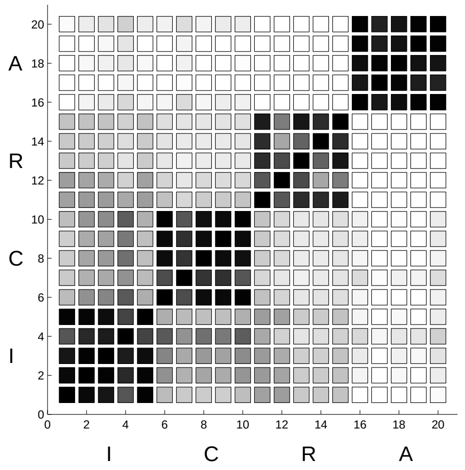

Figure 8: Correlation between the weight vectors of the 20 characters (5 of each letter) fitted by the system. The gray scale is proportional to the correlation, with black corresponding to a correlation of +1 (max. correlation) and white to a correlation of 0 or smaller.

4.3 Experiment 2: Comparison of trajectories in parameter space

An interesting aspect of the CPs is that trajectories with similar velocity profiles tend to have similar w_i parameters, and the CPs can therefore be used to classify movements. To illustrate this, we carried out a simple task of fitting two-dimensional trajectories performed by a human user when drawing two-dimensional single-stroke patterns. Four characters — I, C, R and A — from the Graffiti alphabet used in hand-held computers were chosen. These characters are drawn in a single stroke, and are fed as two trajectories $x(t)$ and $y(t)$ to be fitted by our system. Five examples of each character were presented. Using two sets of 25 parameters w_i (one per dimension), each character was fitted in a single iteration with very little residual error (Figure 7).

Given the temporal and spatial invariance of our policy representation, trajectories that are topologically similar tend to be fit by similar parameters w_i. In particular, computing the correlation $\frac{\mathbf{w}_a^T \mathbf{w}_b}{|\mathbf{w}_a||\mathbf{w}_b|}$ between the parameter

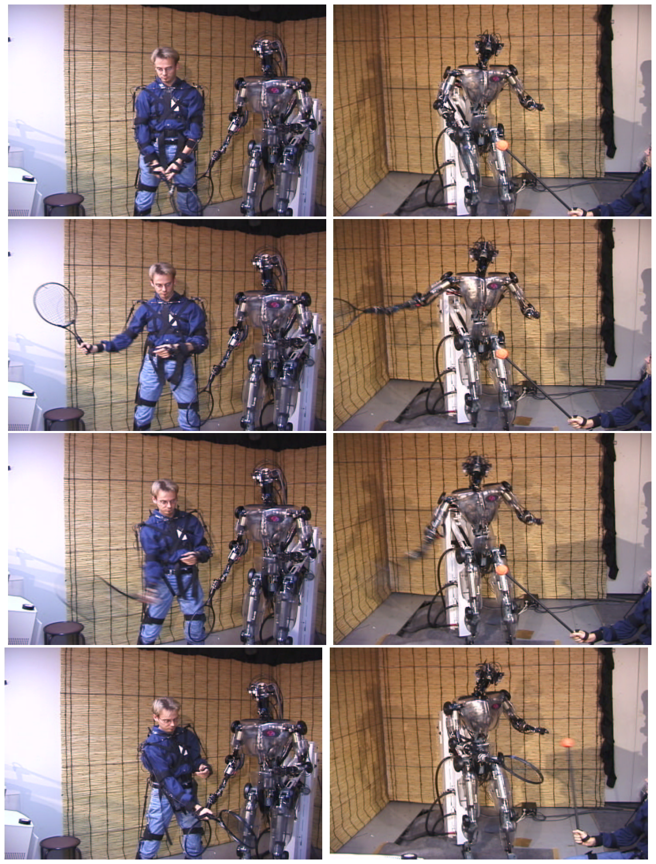
Figure 9: Humanoid robot learning a forehand swing from a human demonstration.

vectors $\mathbf{w}_a$ and $\mathbf{w}_b$ of character a and b can be used to classify movements with similar velocity profiles (Figure 8). In this case, for instance, each of the 20 examples, the correlation is systematically higher with the four other examples of the same character. These similarities in weight space can therefore serve as basis for recognizing demonstrated movements by fitting them and comparing the fitted parameters w_i with those of previously learned policies in memory. In this example, a simple one-nearest-neighbor classifier in weight space would serve the purpose.

4.4 Experiment 3: Learning tennis swings with a humanoid robot

In a last set of experiments, we implemented the system in a humanoid robot with 30 DOFs [5]. The robot is a 1.9-meter tall hydraulic anthropomorphic robot with legs, arms, a jointed torso, and a head. The robot is fixed at the hip, such that it does not require balancing.

The task for the robot was to learn forehand and backhand swings towards a ball demonstrated by a human wearing the joint-angle recording system. The fitted trajectories were fed into an inverse dynamics controller for accurate reproduction by the robot (Figure 9).

Afterwards, the robot was able to repeat the swing motion to different cartesian targets. Using a system of two-cameras, the position of the ball was given to an inverse kinematic algorithm which computes targets in joint space. Similarly to the results of experiment 1, the modified trajectories reached the new ball positions with swing motions very similar to those used for the demonstration. Movies of the experiments can be viewed at http://www-clmc.usc.edu/~ijspeert/humanoid.html.

5 Discussion

We presented a new system for encoding trajectories with control policies (CPs) based on a set of nonlinear dynamical systems. Two key characteristics of the CPs are (1) that movement trajectories are not indexed by time but rather develop out of integrating autonomous nonlinear differential equations, and (2) that the CPs do not encode one single specific desired trajectory but rather a whole attractor landscape in which the desired trajectory is produced during the evolution from the initial states to a unique point attractor, the goal state. This provides the CPs the ability to smoothly recover from perturbations, which is one of the desired properties enumerated in the introduction.

The other desired properties are equally well fulfilled with our system. In particular, the CPs allow fast learning of a trajectory with a relatively low number of parameters. The number of kernel functions (hence of parameters) can be adjusted depending on the desired accuracy of the fit, with more functions allowing the representation of finer details of movement (see [3] for an algorithm to do this automatically). The ease of modification is also ensured by having the goal state of the movement explicitly encoded into the system. The spatial invariance property is used to modulate the trajectory towards the new goal, while keeping the same velocity profile.

We are currently working on extending this work in two directions. First, we are investigating different types of metrics in parameter space for comparing qualitative and/or quantitative features of movements. Second, we are exploring ways to extend the system to encode oscillatory movements in addition to discrete movements.

Acknowledgements
This work was made possible by support from the US National Science Foundation (Awards 9710312 and 0082995), the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and Technology Cooperation, and the ATR Human Information Systems Laboratories.

References

[1] A.J. Ijspeert, J. Nakanishi, and S. Schaal. Trajectory formation for imitation with nonlinear dynamical systems. In *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS2001)*, pages 752–757. 2001.

[2] A.J. Ijspeert, J. Nakanishi, T. Shibata, and S. Schaal. Nonlinear dynamical systems for imitation with humanoid robots. In *Proceedings of the IEEE International Conference on Humanoid Robots*, pages 219–226. 2001.

[3] S. Schaal and C.G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, 1998.

[4] H.K. Khalil. *Nonlinear System.* Prentice Hall, 1996.

[5] C. G. Atkeson, J. Hale, M. Kawato, S. Kotosaka, F. Pollick, M. Riley, S. Schaal, S. Shibata, G. Tevatia, and A. Ude. Using humanoid robots to study human behaviour. *IEEE Intelligent Systems*, 15:46–56, 2000.