Parameterized Temporal Sequences for Motor Control of a Robot System

Karlheinz Hohm,* Torsten Felzer, and Peter Marenbach
Darmstadt University of Technology
Control Systems Theory and Robotics Dept.
Landgraf-Georg-Straße 4, D-64283 Darmstadt, Germany
Phone: ++49-6151-16-7402, Fax: ++49-6151-16-2507
e-mail: hohm@rt.e-technik.th-darmstadt.de
URL: http://www.rt.e-technik.th-darmstadt.de

Abstract. *Learning of temporal sequences is a topic of research in such different areas as speech and other temporal pattern recognition as well as motor control (for a survey see e.g. Mozer, 1993). In this paper an approach is presented which is particularly suitable for motor control due to the fact that it does not only reproduce temporal sequences in exactly the way they where learned but it is able to generate slightly modified sequences according to given parameters, too.*

Key words. Neural Networks, Temporal Sequences, Motor Control, Motor Programs.

1 Introduction

The human motor system is based on the interaction of muscles which are controlled by specific parts of the nervous system. When examining the nervous system, one can recognize a hierarchical structure, physiologically as well as functionally. In the lowest level the spinal cord provides simple motion sequences commonly summarized as reflex actions. They are triggered either by sensory neurons or by signals from the higher level control of the central nervous system. The physically and functionally next level above the spinal cord is the brain stem. There the sensory information from the skin, the muscles and the joints as well as from position of the head gather. It is a kind of intermediate station of most sensory signals going up to the brain as for the control signals going down. In the highest level the motor cortex is the center of motor control. It connects the output of cortical and sub-cortical centers and is the initial point for motor commands going to the brain stem as well as directly to the spinal cord. The translation of motion planning into action is done directly as well as including the basal ganglia and the cerebellum who have to coordinate and control the motor activities. While the motor cortex is responsible for planning and organization of complex motions performed consciously, the cerebellum is the motor controller for timing and precision and therefore also responsible for motions performed unconsciously but started intentionally and thus are no reflex actions (for a detailed model of the human motor system see e.g. Ghez, 1990).

The approach presented here concentrates on the question how the functionality of the intermediate level of the motor system can be imitated to increase autonomy of artificial systems like e.g. robots. Especially the question is examined how the ability of the cerebellum to generate suitable temporal sequences of patterns, to control the above mentioned unconscious motions, can be transferred to artificial learning systems in an appropriate way.

In the next section different methods for generating temporal sequences will be discussed, and in section 3 we will present our approach followed by experimental results in section 4. The paper will conclude with a final discussion of the achieved results and an outlook on future work.

2 Sequence Learning

In most applications of artificial neural networks (ANN) a static mapping of the input to the output pattern is performed. Thus the same input pattern always generates an identical output pattern no matter which inputs were presented to the ANN before. In the context of pattern sequences this static connection is not guaranteed any longer, on the contrary the same input pattern can result in a total different output because the ANN memorizes the context in which the actual input is presented. I.e. even if an identical input pattern is presented to the ANN all the time a varying pattern sequence can occur as output.

*The study reported in this paper was supported by the German Research Foundation (DFG), Graduiertenkolleg "Intelligente Systeme für die Informations- und Automatisierungstechnik"

2.1 Common approaches

To perform such a dynamic mapping it is necessary for the ANN to have recurrent connections. In general ANNs for processing temporal sequences can typically be divided into two functionally different components, a short-term memory with recurrent connections to establish context information and a following long-term memory realized as feed forward net without recurrent connections which performs a static mapping of the current internal state to an output pattern (see e.g. Elman, 1990; Jordan, 1986).

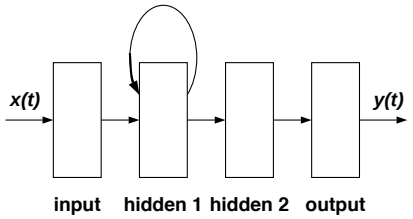

Figure 1: Standard recurrent neural net architecture for sequence processing tasks (see Mozer, 1993).

A survey on how to characterize different realizations of such ANNs suitable for temporal sequence processing, especially how the context information is established, is given by Mozer (1993). There the mainly considered standard architecture of recurrent neural nets as shown in fig. 1 is said not to be the optimal structure because there are problems with training such ANNs with gradient descent methods as back-propagation is and furthermore this kind of ANNs is already failing for relatively simple tasks (Mozer, 1989; Mozer, 1992). Nevertheless it can serve as a basis for new approaches. A slightly modified approach was presented by Stornetta *et al.* (1988). There the weights of the connections between the input layer and the first hidden layer as well as the weights of the recurrent connections are predetermined and fixed. Therefore the problems that arise from using back-propagation to train the remaining weights are avoided. The disadvantage of this method arises from their lower level of flexibility, so that the predetermined weights have to be selected very carefully.

A different architecture was presented by Ans *et al.* (1994). They suggested a network which consists of five modules in each of which the context information is represented within a set of winner-take-all nets. The system has two input layers, one only used during training phase to present the desired sequence elements directly to the output layer and one to characterize the actual sequence by an identifier. The architecture is trained to predict the next sequence element by using its own actual output and the internal context information. Thereby the only input during the recall phase is the sequence identifier, which is kept static during the whole sequence. In this case the long-term memory is realized by the weights between the internal context module and the output module.

The approach presented by Wang and Yuwono (1995) is based on the detection of sub-sequences to predict the next pattern. Therefore a couple of ANNs is trained to detect one characteristic sub-sequence starting within the last few sequence elements and ending with the very last output element of the system. If one of these ANNs detects the sub-sequence it is trained for, it proclaims the next output pattern to be the one associated with itself. All these detectors are arranged in a winner-take-all net to decide which will be the next output. A disadvantage of this system is that the more sequences should be available the more and complicated detectors are necessary to detect definite sub-sequences determining the next output. Thus for learning a new sequence the system itself must be modified. Also for starting a sequence not only the first pattern but an unmistakable start sequence must be presented.

2.2 Motor Programms

A common disadvantage especially but not only of the last two approaches is the fact that those systems are only able to reproduce sequences they have learned. They do not have the ability to generalize between similar sequences as the human brain does in motor control. Considering e.g. a tennis player learning the temporal sequence of a forehand drive, it appears that at first during his first tries – a training phase – the motion is a conscious motion controlled by the cerebellum but supervised by the motor cortex all the time. Later an initial command – like an identifier in (Ans *et al.*, 1994) – given by the motor cortex is enough to recall the whole sequence while the motion itself is performed unconsciously only controlled by lower levels. This is also possible with slight variations of the learned motor sequence since the tennis player is able to hit the ball without conscious control even if it approaches on slightly different trajectories. This principle corresponds to what was described by the term *generalized motor program* by Schmidt (1982). A generalized motor program is a kind of a motor sequence template, which needs certain parameters to define exactly how the sequence is to be executed.

However while a motor sequence is executed conscious interactions by the motor cortex are still possible but due to the time needed for e.g. visual information processing they are restricted by a considerable time delay. From a control theory point of view this corresponds to a stepwise feed-forward control where in between a period of time needed for feed-back control a couple of feed-forward control actions are per-

formed.

3 Computer implementation

Fig. 2 shows the basic architecture of the system called PARAMOUNT for learning and reproducing parameterized motor sequences. This system receives a sequence index (ID) – similar to the ID used by Ans *et al*. (1994) – and a set of parameters from upper (conscious) control levels. The ID is used to specify the motor program to be performed and therefore activates a certain part of the long-term memory. A set of parameters describes which instance of the motor program is desired, e.g. the radius if the motor program to be learned is to draw a circle of different radii.

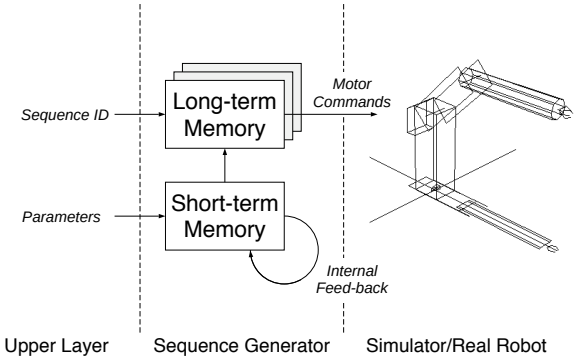

Figure 2: Architecture of a system for learning and reproducing parameterized motor sequences.

In PARAMOUNT each motor program is stored in a different associative memory of the CMAC type (see below) representing the long-term memory. While this CMAC implementation is using a special hash-coding mechanism for the discretisation of its input-space – here the internal state space – a very simple recurrent net structure is used as short-term memory. At each time step the long-term memory produces a seven-dimensional output vector which consists of six components providing movement direction and one overall velocity component. These motor commands are – in the first implementation – interpreted and executed by the robot simulator *Simderella* by van der Smagt (1994), which was adapted to behave as a mantec-r3 robot with six degrees of freedom. After successful experiments with the simulation (see section 4) in a second step PARAMOUNT is now applied to a real robot for further examinations.

3.1 Long-term memory

The *Cerebellar Model Articulation Controller* (CMAC) was first described by Albus (1975) as a – compared to other approaches more exact – model of the cerebellum (cf. fig. 3) and was already successfully applied to wide range of different tasks in learning control (see e.g. Tolle and Ersü, 1992).

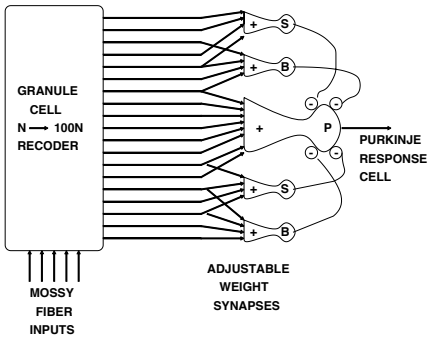

Figure 3: The cerebellum model of Albus.

The association in CMAC is based on a discretisation of the input space where an input vector $\vec{s}$ activates ρ receptors – or association cells. From a mathematical point of view the input space is overlayed with ρ shifted coordinate grids as shown in fig. 4. Each asso-

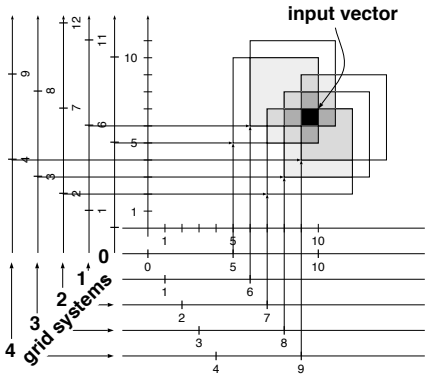

Figure 4: Association in CMAC for a two-dimensional input space and $\rho = 5$.

ciation cell corresponds to one hypercube (square in fig. 4) in one of the grids being its "sensitivity region" and therefore is activated if the input vector is pointing inside of this region.

Within the CMAC implementation used here *hash coding* is used to transform the coordinates of the receptive field into memory addresses to access the synaptic weight related to each activated association cell. The memory output is calculated as the mean value of all activated weights (for details see Mischo, 1996).

3.2 Short-term memory

The input vector of the long-term memory is identically to the output vector of the short-term memory and represents an internal state $\vec{s}$. It consists of the parameters $\vec{p}$ which define the concrete desired mo-

tor sequence and which are kept constant by the short-term memory while the sequence is executed as shown in fig. 5. Additionally to these parameters the vector $\vec{s}$ includes information on the relative position within the sequence. A huge number of possible approaches could be thought of to fulfill this task. The exciting result of a comparison of a few of these approaches was that a very simple one provided the best overall performance. In that approach a single recurrent neuron with constant input of 1 (cf. fig. 5) was applied. This equals a simple counter and can be interpreted as a virtual time base which has to be reset before a new sequence is executed.

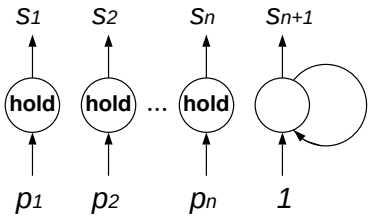

Figure 5: Scheme of the short-term memory; input parameter vector $\vec{p} = (p_1, p_2 \dots p_n)^T$ and output state vector $\vec{s} = (s_1, s_2 \dots s_{n+1})^T$.

The high quality of this very simple approach mainly arises from its combination with the very efficient input coding mechanism of the long-term memory. An advantage of this kind of context information is that interpolation is possible even along the virtual time base. I.e. the long-term memory does not only memorize motor commands at those time steps it was trained. It is also able produce convenient output in between, if the input of the context neuron is set to less than 1.

4 Experimental Results

First successful examinations of PARAMOUNT took place on a UNIX workstation using the robot simulator *Simderella*. As mentioned above the output vectors of the long-term memory are used as motor commands for robot movements. There are two basic possibilities to interpret these values. First they can be used as commands in joint space, i.e. each output vector represents an incremental joint movement. The second possibility is to let PARAMOUNT learn cartesian movements where each output stands for an incremental movements in cartesian space and with the need to transform this output into joint movements using inverse kinematics.

In fig. 6 the drawing of two joint space ellipses is shown. The sequences in fig. 6 were used for training with its height used as parameter p_1. Fig. 7 demonstrates how the system is able to generalize as it produces a trajectory of medium height when a medium p_1 value was applied although it was only trained to recall the sequences shown in fig. 6.

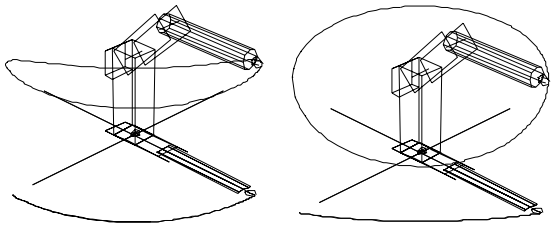

Figure 6: Training of two similar ellipses of different height p_1: $p_1 = 10$ (left) and $p_1 = 45$ (right).

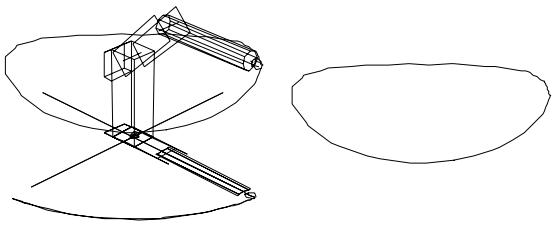

Figure 7: Recalling a generalized ellipse of height $p_1 = 25$: with (left) and without robot (right).

The disadvantage of trajectories specified in joint space is the dependence of their starting point. Thus we did further experiments using the outputs of the long-term memory as motor commands in the cartesian space. Fig. 8 shows a variety of ellipses which

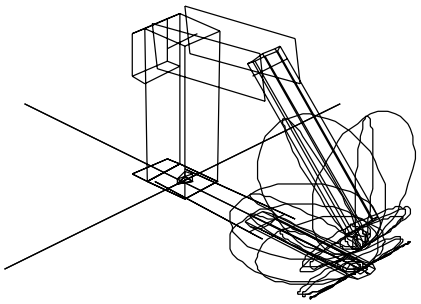

Figure 8: Training set of a cartesian motor program with tree parameters.

were used to define and train a cartesian motor program this time with three parameters: p_1 and p_2 being the height respectively the width of an ellipse and p_3 representing the angle between it and a vertical plane. In several simulations it was shown that after a small number of training cycles – only 10 training cycles consisting of 12 different instances of the same motor program were performed – accurate sequences could be generated for all sets of parameters within the

ranges training took place. E.g. for parameter p_3 0, 45, and 90 degrees appearing in the training set, convenient sequences can be achieved for $p_3 = 0 \ldots 90$.

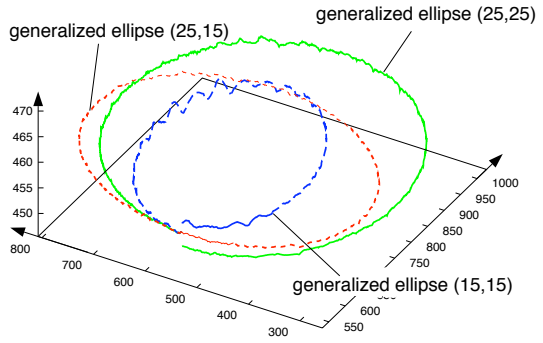

Figure 9: Generalized sequences performed by a real six axes robot (only parameters p_1 and p_2 are given).

Currently the approach is tested on a flexible robot system consisting of a six axes Puma type robot, a PC gathering the sensor information and working as an intermediate station for controlling the robot and the sensors. PARAMOUNT is still running on a UNIX workstation sending its output motor commands to the robot system instead of using *Simderella*. This way the same motor programs that were used in the simulation can be applied to a real robot. Fig. 9 shows three trajectories as they were performed by the robot manipulator when PARAMOUNT was told to produce different instances of the cartesian motor program described above (cf. fig. 8).

5 Conclusions

In this paper a learning approach and its implementation PARAMOUNT for the generation of parameterized temporal motor sequences is presented. As in most other similar approaches a distinction is made between a short-term memory representing an internal state and a long-term memory memorizing the desired output patterns. The important advantage of PARAMOUNT is its capability to generalize, i.e. to generate sequences with slight variations to those which were trained. This is achieved by the concept that a certain motor sequence is seen as an parameterized instance of a class of sequences called a (generalized) motor program. PARAMOUNT was successfully examined using the robot simulator *Simderella* and is currently applied to a real six axes robot system.

References

Albus, J. (1975). A new approach to manipulator control: The cerebellar model articulation controller. *Transactions ASME*.

Ans, B., Coiton, Y., Gilhodes, J.-C. and Velay, J.-L. (1994). A neural network model for temporal sequence learning and motor programming. *Neural Networks* **7**(9).

Elman, J. L. (1990). Finding structure in time. *Cognitive Science* **14**.

Ghez, C. (1990). The control of movement. In: *Principles of Neural Science* (E.R. Kandel and J.H. Schwartz, Eds.). 3 ed. Elsevier Science.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In: *Proc. 8th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ.

Mischo, W. S. (1996). How to adapt in neurocontrol: A decision for CMAC. In: *Neural Adaptive Control Theory* (R. Zbikowski and K. J. Hunt, Eds.). World Scientific.

Mozer, M. C. (1989). A focused back-propagation algorithm for temporal pattern recognition. *Cognitive Science* **3**.

Mozer, M. C. (1992). The introduction of multiscale temporal structure. In: *Advantages in neural information processing systems 4*. San Mateo, CA.

Mozer, M. C. (1993). Neural net architectures for temporal sequence processing. In: *Predicting the Future and Understanding the Past* (A. Weigend and N. Gershenfeld, Eds.). Addison-Wesley Publishing.

Schmidt, R. A. (1982). *Motor Control and Learning – A Behavioral Emphasis*. Human Kinetics Publishers. Champaign, Illinois.

Stornetta, W., Hogg, T. and Hubermann, B. (1988). A dynamical approach to temporal pattern processing. In: *Proc. IEEE Conf. Neural Information Processing Systems*. Denver, CO.

Tolle, H. and Ersü, E. (1992). *Neurocontrol*. Vol. 172 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag. Berlin.

van der Smagt, P. (1994). Simderella: A robot simulator for neurocontroller design. *Neurocomputing*.

Wang, D. and Yuwono, B. (1995). Anticipation-based temporal pattern generation. *IEEE Trans. Systems, Man and Cybernetics* **25**(4).