Motor Sequence Processing with an Artificial Learning System

Torsten Felzer[1], Peter Hartmann[1],
Karlheinz Hohm[2], and Peter Marenbach[2]

[1] Technical University of Darmstadt, Department of Computer Science
Alexanderstr. 10, D-64283 Darmstadt, Germany
E-Mail: {felzer,hartmann}@isa.informatik.th-darmstadt.de
[2] Technical University of Darmstadt, Institute of Control Engineering
Landgraf-Georg-Str. 4, D-64283 Darmstadt, Germany
E-Mail: {hohm,mali}@rt.e-technik.th-darmstadt.de

Abstract. This paper describes a neural-based artificial system, called ParaMount, for learning, storing, and reproducing parameterized motor sequences, i.e. temporal sequences consisting of motor commands as sequence elements. ParaMount is designed to control the arm of a robot, mimicking the functionality of specific parts of the human motor system up to a certain extent. After having been trained with a number of sample sequences, ParaMount is not only able to recall these sequences exactly as they were learned, but also to generalize slightly modified (scaled) versions according to a set of parameters.

1 Introduction

The process of generating complex movements (or *motor sequences*) in humans can be explained on the basis of a three-step hierarchy. *Conscious* motion planning and the triggering of the selected motor sequences take place at the highest level. The intermediate *control level* reacts by reproducing a memorized sequence of elementary movements, which is supervised and controlled unconsciously. This finally results in the contraction of individual muscle fibers at the lowest *execution level*.

Humans are able to perform complex movements faster than it is possible to follow the trajectory consciously, due to the long time needed for processing sensory feedback. This argument is one aspect leading to the belief that entire motor sequences are stored in an appropriate way, so that reproducing them is comparable to looking up the elementary movements in a table, i.e. can be done very fast.

The total number of different motor sequences is extremely large, since, for example, even a tiny variation of the velocity of the elementary movements yields actually different motor sequences. Due to the (huge but) finite storage capacity of the human brain, it is evident that not every single motor sequence can be stored separately. Instead, representatives of clusters of scaled motor sequences are memorized as "meta-sequences" that can be reproduced in a parameterized way. This principle corresponds to the notion of *generalized motor programs* used by Schmidt (see [11]).

Recalling a certain sequence thus requires specifying the motor program the sequence belongs to and the parameters defining this very instance of the meta-sequence. An initial command encoding this information given by the motor cortex suffices to generate the entire sequence.

This paper introduces a neural-based artificial system that mimics the functionality of the intermediate level – based on the concept of motor programs – in order to control the arm of a robot by producing a sequence of motor commands. The system's most important feature is its ability to generalize sequences it has never "seen" before – *scaled* by a set of parameters – from a number of learned sample sequences. Because the system is used to store and to generate parameterized motor sequences, it has been given the name ParaMount.

The next section provides a short survey of common approaches for temporal sequence processing. Section 3 is devoted to the implementation of the ParaMount system, i.e. a detailed description of its architecture, while section 4 deals with the results obtained in several simulation runs. The paper concludes with a brief summary and a discussion of future work.

2 Related Work

A typical artificial neural network (ANN) implements a (similarity-preserving) mapping from an input onto an output pattern. This paper concentrates on the processing of time-varying patterns, so-called *temporal sequences*. In this case, the mapping from input to output patterns depends on the temporal context.

There are a lot of different approaches as for the design of such an ANN, the basic structure of which is depicted in figure 1. The network comprises two functionally different components: a short-term memory (STM) – calculating some representation of a temporal *state* of the system with the help of recurrent connections – and a long-term memory (LTM) which performs a static mapping of this state representation onto the corresponding output pattern (see, e.g., [3], [6]).

Different architectures especially for realizing the STM are presented in [9], and the major problem here is the training of the recurrent connections. Adjusting the weights with the help of a gradient descent procedure, such as back-propagation (see [10]), leads to serious problems if the system has to retain past information over a long time span (see [8]).

The approach of Ans et al. (see [2]) differs from back-propagation networks in that it consists of five layers of artificial neurons with winner-take-all inter-

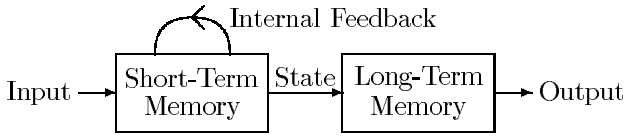


Fig. 1. ANN for temporal sequence processing (adapted from [9])

connections. Each layer stores a representation of some discrete internal state. When learning a sequence, the system is trained to predict, at each time step, the next sequence element based on the current temporal state. Recalling a learned sequence can be triggered by specifying a sequence identifier, which has been associated to the sequence during the learning phase. Coupled with a sensorimotor system (with two degrees of freedom) the network is able to learn and recall simple motor sequences. Its major disadvantage, however, in contrast to ParaMount (see next section), is its inability to generalize among similar sequences, i.e. to recall them in a parameterized way.

3 Architecture of the ParaMount System

The basic architecture of the ParaMount system and the connections to the conscious and the execution levels of the hierarchy mentioned in the introduction are shown in figure 2, the individual components of which are explained in the following subsections.

3.1 The User

The user of the ParaMount system plays the role of the conscious level within the human motor system. He is responsible for specifying the training sequences during the initial learning phase, and a sequence ID[3] and the parameters in the recall phase, respectively.

For the specification of temporal sequences, the auxiliary tool Spemola[4] has been developed, which represents a compiler for a simple programming language allowing to create descriptions of complex sequences in an intuitively intelligible way.

3.2 The Long-Term Memory

The function of ParaMount is comparable to that of the cerebellum in the human motor system. Albus (see [1]) proposed the *Cerebellar Model Articula-*

[3] The ID is an abstract "name" for a motor program, similar to the sequence identifier in the approach of [2].

[4] Spemola is an acronym for "Specification language for describing motor sequences".

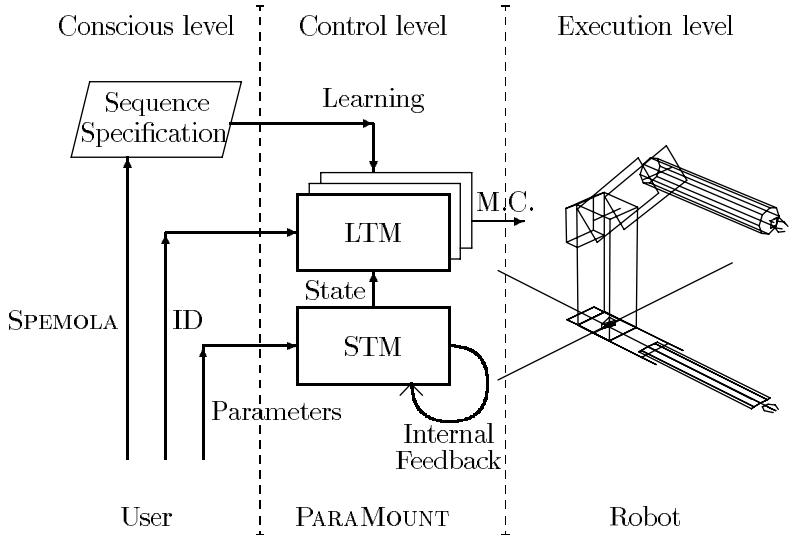


Fig. 2. Architecture of PARAMOUNT (M.C. ≡ "Motor Commands")

tion Controller (CMAC) as a theoretical model of the cerebellum. Its computer-oriented effective implementation AMS[5] (e.g. [7], [4]) therefore was a sort of "natural choice" for the LTM module in PARAMOUNT.

Like most ANN models, the AMS maps similar input patterns onto similar output patterns and is therefore interpolating when presented with an untrained input. Unlike typical back-propagation networks, the AMS makes use of a *local* weight update scheme in the learning phase and therefore converges very fast. The association is based on the following principle. An input pattern $\mathbf{i}$ activates a set of output cells, each storing one specific value. The weighted average of all these values is the overall output o of the system when presented with input pattern $\mathbf{i}$. The fact that the sets of cells activated by "similar" input patterns overlap is responsible for the ability of the AMS to interpolate. In PARAMOUNT, each motor program is stored in an own AMS module meaning that the sequence ID serves as a selector among multiple LTM modules. The association performed by the AMS is purely static, so for the generalization of *sequences* rather than single patterns, this module has to be combined with an STM.

3.3 The Short-Term Memory

The output of the STM is an internal state vector $\mathbf{s}$. This vector, which serves as input to the LTM, has to provide two kinds of information. First, information on the parameter set (given as a vector $\mathbf{p}$), and second, information about the

[5] AMS stands for "Associative Memory System".

relative position within the sequence. There is a huge number of possible ways how this state vector can be calculated, and a variety of recurrent neural networks has been tested as design alternatives for the STM. The remarkable result of a comparison of these approaches was that the very simple one illustrated in figure 3, using only one single recurrent neuron, yielded the best overall performance.

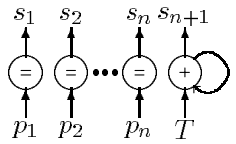


Fig. 3. Schematic sketch of PARAMOUNT's STM module with parameter vector $\mathbf{p} = (p_1, p_2, ..., p_n)^{\mathrm{T}}$, state vector $\mathbf{s} = (s_1, s_2, ..., s_{n+1})^{\mathrm{T}}$, and time constant T.

The value of s_{n+1} represents a counter that is set to 0 each time the processing of a new sequence is started and incremented by T at each time step. This simple STM together with the interpolating AMS constitute a sort of ideal combination for the intended application. The input encoding mechanism of the AMS makes it even possible to generalize along the virtual time axis (realized by the counter) upon variation of the time constant T between learning and recalling a sequence.

3.4 The Robot

The output of the LTM is a seven-dimensional vector comprising six direction components and one overall velocity component. The outputs are sent, in the form of motor commands, to a robot which then executes the elementary movements.

In the first implementation, the robot simulator *Simderella* by v. d. Smagt (see [12]) – behaving as a six axes Puma type robot – has been used to visualize the motor sequences on a computer display.

4 Simulation Results

First successful experiments with PARAMOUNT took place on a UNIX workstation. For any given motor program, the operation of the system can always be divided into two phases: a learning phase – where a number of sample sequences are stored in the associative memory – and a recall phase – where various sequences are reproduced with different parameter sets.

One of the first motor programs to be examined is visualized in figure 4. The task was to learn how to draw the small script 'a' in different sizes. Although the system was only trained to recall the leftmost and the rightmost sequences of figure 5A, it was also able to generate all sequences in between. Figure 5B

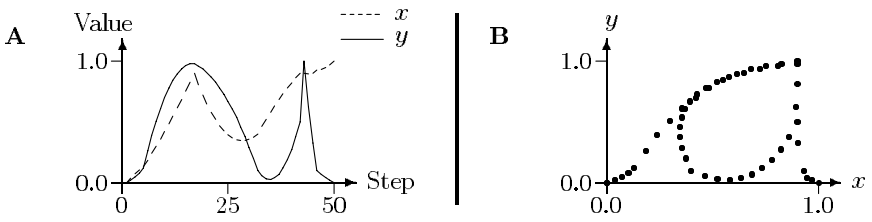


Fig. 4. Script 'a' as two-dimensional sequence – A: x and y drawn with respect to the sequence steps; B: y drawn with respect to x.

demonstrates that the deviation from the training sequences diminished exponentially with an increasing number of training cycles. In order to evaluate the generalization (and reproduction) performance of the system, the five sequences represented in figure 5A were compared with their respective *reference sequences*, i.e. the target output patterns. Table 1 lists the average relative error for the x and y dimensions.

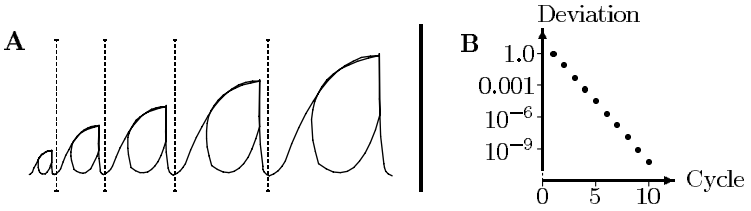


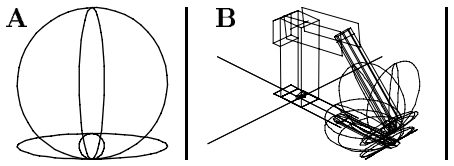
Fig. 5. A: 'a' in different sizes with p_1 as "size-parameter" – leftmost ($p_1 = 1.0$) and rightmost ($p_1 = 5.0$) sequences are learned, the others ($p_1 = 2.0, 3.0, 4.0$) are generalized; B: Evolution of the error during training (logarithmic scale!).

A more abstract and also more complicated motor program (using 3 parameters) is illustrated in figure 6. p_1 and p_2 determine an ellipse's width and height, respectively, and p_3 defines the angle between the ellipse and a vertical plane. Figure 6A shows the 4 combinations of the smallest (5.0) and largest (30.0) values for height and width, and 6B depicts these 4 ellipses drawn at 3 different angles each.

Table 1: Relative deviation of the 5 recalled sequences of figure 5A from their respective reference sequences (on the average).

Relative Percentage	**Size 1.0**	**Size 2.0**	**Size 3.0**	**Size 4.0**	**Size 5.0**
x-Dimension	0.0	1.7	5.5	2.1	0.0
y-Dimension	0.0	1.4	5.7	2.3	0.0

This second example demonstrates again very impressively how powerful the PARAMOUNT system is and how fast it converges. After only 10 training cycles through the set of 12 ellipses, the system was able to generate accurate ellipses (not more than 10.0 % deviation from reference sequences – for an example see 6C) belonging to almost arbitrary sets of parameters, the only restriction being that the parameters had to be chosen within the training interval[6].



	%
x	1.1
y	10.0
z	7.9

Fig. 6. A and B: Training set of an ellipse-drawing motor program with three parameters; C: Relative deviation from reference sequence for a generalized ellipse with $p_1 = p_2 = 20.0$ and $p_3 = 30.0$.

Several different motor programs have been tested using a wide range of LTM configurations in PARAMOUNT. After some search effort (and with some experience), it was always possible to find a set of configurations that led to fast convergence while resulting in a relatively small generalization error.

5 Conclusion

The neural-based system PARAMOUNT has been described, which is able to record temporal sequences and to recall them taking into account a set of parameters. The sequence elements are translated into motor commands which control the arm of a computer-simulated robot. Clusters of sequences, each one being a scaled version of the same basic sequence, are associated with a unique identifier each. The identifiers correspond to abstract names of motor programs, such as "draw circle", and the radius of the circle is specified upon the variation of the parameters. Results reported here have demonstrated that the system can generalize sequences belonging to parameters which have not been learned explicitly – with only moderate deviations from reference sequences – based on a relatively small training set.

The first promising results presented here may be seen as a starting point for future developments in this area. The foremost task is to examine the coupling of PARAMOUNT with a real instead of a simulated robot and all questions related to that, especially as far as optimization of learned sequences is concerned. First steps in this direction have already been taken (see [5]). An application pointing to a much further future is linked to helping people suffering from a motor impairment by supplementing (unconscious parts of) their nervous system with a sort of "neuro-prosthesis" based on a system like PARAMOUNT.

[6] For instance, the values 0.0, 45.0, and 90.0 appeared in the training set for parameter p_3; therefore p_3 had to be chosen from the range 0..90 at recall time.

References

1. Albus, J. S.: A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Transactions ASME* (1975)
2. Ans, B., Coiton, Y., Gilhodes, J.-C., Velay, J.-L.: A Neural Network Model for Temporal Sequence Learning and Motor Programming. *Neural Networks (Elsevier Science)* 7(9) (1994) 1461–1476
3. Elman, J.: Finding structure in time. *Cognitive Science 14* (1990) 179–211
4. Ersü, E., Tolle, H.: Hierarchical learning control – an approach with neuron-like associative memories. In *IEEE Conference on Neural Information Processing Systems - Natural and Synthetic*, Denver, CO (1987) pp. 249–261
5. Hohm, K., Felzer, T., Marenbach, P.: Parameterized temporal sequences for motor control of a robot system. MSCA'96, Crete, Greece (1996)
6. Jordan, M. I.: Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ (1986) pp. 531–546
7. Mischo, W. S.: How to adapt in neurocontrol: A decision for CMAC. In Zbikowsky, R., Hunt, K. J. (Eds.), *Neural Adaptive Control Theory*. World Scientific (1996)
8. Mozer, M. C.: The induction of multiscale temporal structure. In Moody. J. E., Hanson, S. J., Lippmann, R. P. (Eds.), *Advances in Neural Information Processing Systems 4* (1992) pp. 275–282. San Mateo, CA: Morgan Kaufmann.
9. Mozer, M. C.: Neural net architectures for temporal sequence processing. In Weigend, A. S., Gershenfeld, N. A. (Eds.), *Time series prediction: Forecasting the future and understanding the past*, Reading, MA: Addison-Wesley (1993)
10. Rumelhart, D. E., Hinton, G. E., Williams R. J.: Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Volume 1 (1986) pp. 318–362. Cambridge, MA: MIT Press
11. Schmidt, R. A.: *Motor control and learning: A behavioral emphasis*. Champaign, Ill.: Human Kinetics Publishers (1982)
12. v. d. Smagt, P.: Simderella: a robot simulator for neuro-controller design. *Neurocomputing (Elsevier Science) 6*(2) (1994)

This article was processed using the LaTeX macro package with LLNCS style