Training networks of biological realistic spiking neurons for real-time robot control

Harald Burgsteiner

harald.burgsteiner@fh-joanneum.at

InfoMed / Health Care Engineering
Graz University of Applied Sciences
Eggenberger Allee 9-11
A-8020 Graz, Austria

Abstract

We present a framework for learning to control robots in real-time with networks of biologically realistic spiking neurons. The method combines a real-world device that operates in real-time, controlled through a simulated recurrent spiking neural network which is randomly generated. Learning does not take place inside the network, only the weights of the output units are changed during training. It will be shown, that it is possible to use such simple types of biological realistic spiking neural network – also known as a *neural microcircuits* – to learn at least reactive control behaviors like that incorporated in Braitenberg vehicles and additionally also non-linear controller types. Furthermore, it has already been shown in [1] that tasks that involve temporal integration of sensory information can be accomplished. The results suggest that a neural microcircuit with a simple learning rule can be used as a sustainable robot controller for experiments in computational motor control.

Keywords: real-time, robot control, spiking neural networks, learning method, prediction.

1 Introduction

There are many reasons for using neural networks for robot control. Some of them are to be able to do information processing of real-world sensors in real-time, learning of complex tasks on a variety of time scales and precise controlling of actuators. Many successful approaches that also use a sort of "learning" have been found to accomplish some of these tasks. Some of that are (i) reinforcement learning techniques that reward or punish a subject for good or bad actions respectively, (ii) genetic algorithms [2] that "evolve" robots over many generations, (iii) echo-state networks [3] that incorporate recurrent artificial neural networks and finally also (iv) approaches based on biologically realistic neural networks (i.e. "spiking neural networks"). The latter approach seems to be one of the most interesting, since this would include to be able to understand parts of the information processing that occurs in biological creatures.

More complex robot controllers have to incorporate state information in the processing. When not all state information can be derived from the current sensory information, the state information has to kept in some kind of memory within the controller. A common way to achieve short term memory for state information in neural networks is to use recurrent neural networks. Figure 1 sketches the possible flow of information within a strict feed-forward neural network in comparison to a network with recurrencies. Recurrent neural networks represent a non-linear dynamical system with a high-dimensional internal state,

which is driven by the input and – due to the recurrency – also from the current output. Hence, a history of (recent) inputs is preserved in such a network. This was first noticed by [4] who developed a network with a specialized architecture to provide "long short-term memory".

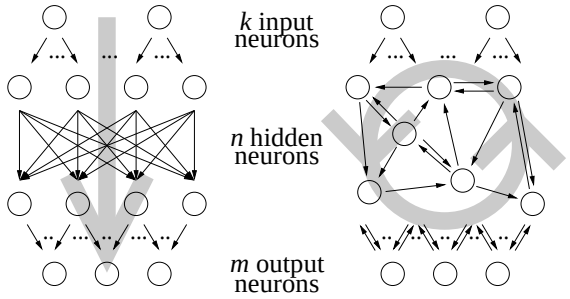

Figure 1: *Comparison of the architectures of a feed-forward (left hand side) with a recurrent neural network (right hand side); the grey arrows sketch the possible direction of computation*

These feature allows a system to incorporate a much richer range of dynamical behaviors. Many approaches have been elaborated on recurrent artificial neural networks (ANNs). Some of them are (i) dynamic recurrent neural networks (with backpropagation through time, see [5]), (ii) radial basis function networks [6] (in the strict sense that networks with lateral connection can also been seen as recurrent), (iii) Elman networks (recurrency through a feedback from the hidden layer onto itself, see [7]), (iv) self-organizing maps, (v) Hopfield nets and (vi) the "echo state" approach from [3]. In case of autonomous agents it is rather difficult to employ strictly supervised learning algorithms for recurrent ANN such as backpropagation, Boltzmann machines or Learning Vector Quantization (LVQ), because the correct output is not always available or computable. A different method is incorporated in "echo state networks" from [3], which are in principle recurrent artificial neural networks like the one shown on the right hand side of Figure 1: k input units project arbitrarily onto a recurrent network of n units. These n units form the "dynamic reservoir" from which the desired m outputs are combined. The basic idea behind learning in echo state networks (and the Liquid State Machine which will be described later) is, that one does *not* try to set the weights of the connections within the pool of neurons but instead reduces learning to setting the weights of the readout neurons. This reduces learning dramatically and much simpler supervised learning algorithms which e.g. minimize the mean square error in relation to a desired output can be applied. No specialized gradient descent based algorithms are necessary and one of the many available linear regression algorithms can be used. The weights for the connections within the dynamic reservoir are chosen at the beginning from a random distribution.

2 Liquid State Machines

A disadvantage of spiking neural networks in comparison to ANN is, that due to the non-continuous output function computed by such neuron models, standard learning algorithms based on gradient descent methods do not apply. Attempts to modify backpropagation and other methods have yielded little success. Hence, learning rules have so far concentrated mainly on unsupervised adaptation rules like Hebbian learning and other non-associative dynamic synaptic effects. A new framework for computing and learning in spiking neural networks called "liquid state machine" (LSM) was introduced in [8]. The term "liquid state" refers to the idea to view the result of a computation of a neural microcircuit not as a stable state like an attractor that is reached. Instead, a neural microcircuit is used as an *online computation tool* which receives a continuous input that drives the state of the neural microcircuit. The result of a computation is again a continuous output generated by readout neurons given the current state of the neural microcircuit. The main difference to echo state networks is, that the LSM is based on a pool of biological realistic spiking neurons instead of the recurrent artificial neural network used by the echo state networks. Figure 2 shows an example network architecture of a liquid state machine as it was used in our robot controller experiments. The

recurrency also enables LSMs to solve tasks that require temporal integration. This has already been successfully shown in the field of robotics with the task of movement prediction in the visual field of a robot [1] where a LSM had to predict the position of a ball a certain time in the future. The idea of reducing learning to setting the weights of the readout neurons only, allows much simpler supervised learning algorithms – which e.g. only have to minimize the mean square error in relation to a desired output – to be applied.

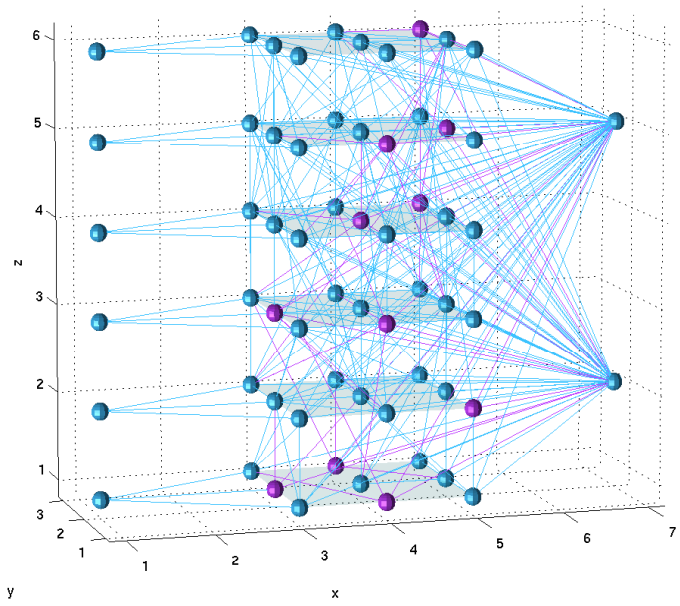

Figure 2: *A LSM as it is used in the imitation learning experiments: the 3x3x6 spiking neurons of the LSM are placed on a regular 3D grid. Additionally, 6 linear input neurons are used for sensor input (left hand side) and 2 linear output neurons are used for motor output. All randomly generated connections for* $\lambda = 2$ *are shown*

2.1 Neural microcircuit

The model of a neural microcircuit as it is used in the LSM is based on evidence found in [9] and [10]. It is the biggest computational element within the LSM, although multiple neural microcircuits could be placed within a virtual model. In a model of a neural microcircuit $N = n_x \cdot n_y \cdot n_z$ neurons are placed on a regular grid in 3D space. The number of neurons along the x, y and z axis, n_x, n_y and n_z respectively, can be chosen freely. A further factor determines how many of the N neurons should be inhibitory. Another important parameter in the definition of a neural microcircuit is the parameter λ, because number and range of the connections between the N neurons within the LSM depend upon it. The probability of a connection between any two neurons i and j is given by $p_{(i,j)} = C \cdot e^{-D_{(i,j)}/\lambda^2}$ where $D_{(i,j)}$ is the euclidian distance between the two neurons and C is a parameter depending on the type (excitatory or inhibitory) of each of the two connected neurons. In our experiments we used spiking neurons according to the standard leaky-integrate-and-fire (LIF) neuron model that are connected via dynamic synapses. The time course for a postsynaptic current is approximated by the equation $v(t) = A_k \cdot e^{-t/\tau_{syn}}$ where A_k is a synaptic weight and τ_{syn} is the synaptic time constant. The "weight" A_k of a dynamic synapse depends on the history of the spikes it has seen so far. It is determined according to the model from [11], where A_k depends on 3 parameters U, D and F, where U is the utilization of the synaptic efficacy, D and F are the time constants that model the recovery from depression and facilitation respectively.

2.2 Real-time extensions of the Liquid State Machine

The simulator for the LSM which is located at `http://www.lsm.tugraz.at` is a tool that was mainly designed for fast offline experiments, i.e. pre-recorded or synthetic data is used as input to the network and results are written to files that can be analyzed afterwards. We expanded the possibilities of the simulator to do real-time experiments. This extension allows to use data from arbitrary real-world sensors in real-time as inputs to the LSM. The specialized core of the LSM software computes the network's dynamics. Any output of the network can then again be applied to real-world actuators, like motors that drive a robot in an experimental environment.

For the standard LSM a connection between the time passing in the simulation and the time passing meanwhile in the real world is not necessary. This assumption does not hold anymore for robot experiments that incorporate real-world devices. Since the training of the readout neurons of a LSM is based on supervised learning algorithms one requires not only input data but also an appropriate target function. In e.g. [12], [13] and [8] the target vectors have been calculated in advance and the input data at a time t_2 is not dependent from any output values at a time $t_1 < t_2$, i.e. no feedback is in use. Hence, the state vector of the LSM $x(t) := f(u(\cdot))$ and the output is defined as $f_1(x(t))$. This mode of operation is often called "open loop" control system, with applications in e.g. speech recognition (see [12]). Additionally, it is often not possible to calculate a target functions for an experiment in the real-world because a simulation of the whole environment would be computationally too complex. It enriches the set of possible input functions if one uses real-world sensory data as input to simulated robot controls. Since the controller influences these sensory readings, the timing of the real-world has to be considered in the simulations. This can be efficiently done with the real-time extensions of the LSM.

3 Experimental Setup

A framework consisting of the RT-LSM and a standard miniature robot called Khepera was used to implement the experimental setup. A Khepera is first steered by a programmed controller in a precisely defined environment that can be seen in Figure 3. All infra-red sensor and motor speed data are recorded. These data are presented to a spiking neural network simulated by the LSM. The goal is to imitate and generalize the behavior by a controller now consisting of a trained spiking neural network (more exactly by trained read-out neurons of the network). The difficulty is, that the original predefined behavior is only available in the form of recorded raw data and not in form of any rules that could be extracted out of the previously used program. The simulated controller itself consists of a single column of a liquid state machine. The weights that are connecting some spiking neurons from the neural microcircuit to the linear output units are the only parameters that are trained during the learning phase. These output units are directly connected to the motor units of the Khepera. The input to this neural microcircuit consists solely of the values that are read from the frontal sensors of the Khepera. Hence, the continuous input to the neural microcircuit is determined by the movement of the robot, which in turn is controlled by the neural microcircuit. The robot was first controlled by two different algorithms, that were programmed to display an obstacle avoidance behavior.

3.1 Controller types

For the first experiments in robot control with a neural microcircuit the master models were restricted to simple reactive controllers. One of the controllers used in the experiments was a Braitenberg type controller with obstacle avoidance behavior. Such controllers originally consist of two perceptrons with sigmoidal output functions fully connected with Khepera's 6 frontal infrared proximity sensors. A robot with such types of controllers can only move in a forward direction. The outputs of the two perceptrons are directly "wired" with the two motors. The 12 synaptic weights determining the share of each of the 6 frontal sensor to the 2 output neurons are set to, e.g. {0.3, -0.3}, {0.6, -0.6}, {1, -1}, {-1, 1}, {-0.6, 0.6} and {-0.3, 0.3} respectively. Each weight vector contains the synaptic strength for the connection to the left and to the right motor neuron. Additionally, the two neurons both get a small positive bias to provide the robot with a constant forward movement of about 5 cm per second for the case that no obstacles can be detected within

the range of the proximity sensors. The values delivered by the 6 frontal proximity sensors are normalized in the range $[0,1]$ where a value of 0 indicates no obstacle in the range and 1 denotes a very close obstacle.

In a second set of experiments the Braitenberg controller was replaced with a non-linear reactive controller. Here, the controller chooses between 3 types of movements: (i) straight forward movement when no collision can be detected on any of the sensors. In this case, the motor speeds are set to values of $\{5, 5\}$, producing a speed of about 5 cm per second. (ii) a turn to the right on its location, when an obstacle is detected with one of the left 3 sensors, resulting in a speed vector of $\{5, -5\}$ and (iii) a turn to the left, when an obstacle is detected with one of the sensors on the right hand side with motor speeds of $\{-5, 5\}$. A collision is defined as a sensor value crossing a threshold of about 90% of its maximum value. Initial experiments suggested this threshold to be reasonable. In case of a collision, the turns are carried out until all sensors report no further collision. In case of a full frontal collision where an obstacle is detected on both sides, a turn to the left was arbitrary chosen to have a higher priority.

3.2 Creation of training and test data

In the first phase of an experiment, sensor and motor data are collected while the robot is controlled with one of the previously described controllers. The collected data is then partitioned into segments with a single occurrence of an obstacle. These episodes are stored individually in a database. The beginning of an episode is defined to be about 100 ms before the occurrence of an obstacle at any one of the 6 sensors, after no obstacle could be detected for at least 100 ms. The end of an episode is marked when all sensors signal that the obstacle is no longer in their range (i.e. when the values of all 6 frontal IR proximity sensors decrease beneath a threshold of about 2% of their maximum value which is beyond the usual noise level).

During the training phase, arbitrary episodes are selected from the database containing all episodes. Additionally, quiet phases with random lengths are inserted between every two episodes. This way, large amounts of different training data can be generated, without having to construct different environments after each training run. After the training phase, the fitness of the network is tested with several more episodes again with random pauses in between. The Khepera is then controlled by the spiking neural network in a real-time mode. Further details for the experimental setup can be found in [14].

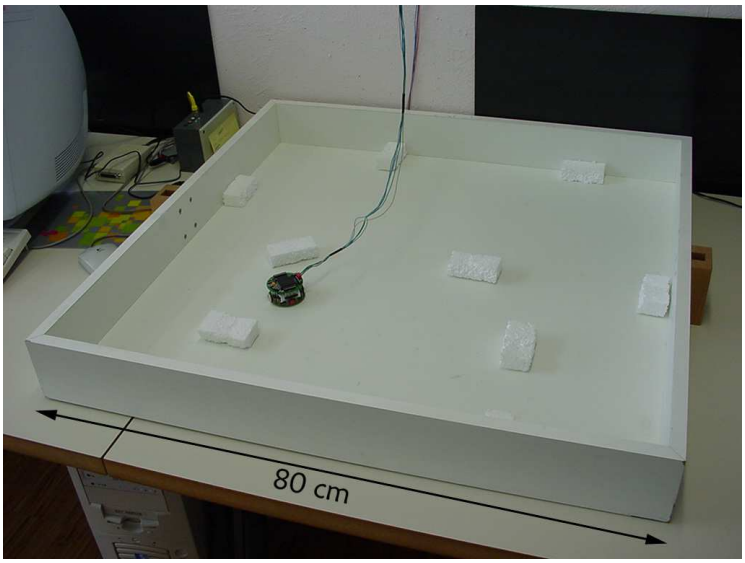

Figure 3: *A typical setup as it was used during our experiments: the Khepera is running within an area of about 80x80 cm with well reflecting obstacles and walls. It is connected to the controlling PC via a cable with rotating contacts to supply it with continuous power to enable long term experiments. The obstacles were randomly placed before every training run*

3.3 Parameter settings of the neural microcircuit

Figure 2 shows a typical neural microcircuit as it is used in our experiments. The neural microcircuit consists of 54 leaky integrate-and-fire neurons arranged on the regular grid points of a 3x3x6 cube in 3D. 20% of these neurons were randomly chosen to be inhibitory. The rest consisted of excitatory neurons. The probability of a synaptic connection between any two intra-column neurons is modeled as $p_{(i,j)} = C \cdot exp^{-\frac{D_{(i,j)}}{\lambda^2}}$ (see section 2.1 for details). This way, the probability distribution is chosen to prefer local connections. In our experiments we set $\lambda = 2$. The remaining parameter C of the equation is depending on the types – excitatory (E) or inhibitory (I) – of the connected neurons: $C_{EE} = 0.3$, $C_{EI} = 0.4$, $C_{IE} = 0.2$ and $C_{II} = 0.1$. These values are chosen according to [9]. The parameters determining the dynamical behavior of the intra-column neurons and synapses were chosen from data given in [9] and [11]. The membrane time constant of all neurons is 30 ms, the absolute refractory period is set to 3 ms for excitatory neurons and 2 ms for all inhibitory neurons. Presuming a membrane resting potential of 0 Volts, the threshold of each neuron is set to 15 mV and the reset voltage for each neuron is drawn from a uniform distribution in the interval $[13.8\,\text{mV}, 14.5\,\text{mV}]$. Additionally, a non-specific background current I_b is again drawn from a uniform distribution in an interval $[13.5\,\text{mA}, 14.5\,\text{mA}]$ for each individual neuron.

The synapses connecting the neurons inside the column are modeled with short-time dynamics according to [11], where the parameters U, D and F determine the utilization of the synaptic efficacy, the time constant of depression (in [s]) and the time constant for facilitation (in [s]), respectively. The actual parameters are set to values chosen from a Gaussian distribution with a standard deviation of 50% of its mean. The mean values again depend on the types of the neurons which a particular synapse connects. For EE-type connections the mean values for U, D and F are 0.5, 1.1 and 0.05. IE-type connections have mean values of 0.05, 0.125 and 1.2. EI-type connections are modeled with mean values of 0.25, 0.7 and 0.02. Finally, the dynamical parameters U, D and F for II-type connections have mean values of 0.32, 0.144 and 0.06. The postsynaptic currents of all synapses are modeled with a simple exponential decay $e^{-\frac{t}{\tau_s}}$ with $\tau_s = 3\,\text{ms}$ for excitatory and $\tau_s = 6\,\text{ms}$ for inhibitory synapses. The transmission delays between every two liquid neurons were uniformly chosen to be 1.5 ms for EE-type connections and 0.8 ms for all other connections.

The values of the 6 frontal IR proximity sensors have to be fed continously into the liquid. Therefore, 6 linear input neurons with the ability to receive external input in real-time were used. These types of neurons can inject a current directly into postsynaptic neurons. In our experiments, static analog synapses were used to connect the excitatory input neurons to specific neurons in the neural microcircuit. We used a simple form of spatial coding to connect the input neurons to the column. As it is illustrated in Figure 2 each input neuron projects its input current onto 3 unique neurons placed at the most frontal layer of the column.

4 Results

The basic task in the experiments was to teach the generic neural microcircuit described in the previous sections to imitate a behavior that was demonstrated and recorded by the controller architectures introduced in the previous section. During the phase of collecting real-world data – when Khepera was controlled by the Braitenberg type vehicle – $n = 1859$ encounters with obstacles were recorded and fed into the database. For each run of the experiment, 500 of those episodes are randomly chosen to be training examples and 125 more are randomly saved for the later use as test data. The training episodes are then connected into a single long data stream with arbitrary pauses between each two obstacle encounters. The maximum pause that can occur was set to be 25 seconds. The same is valid for the test data stream. Using this technique to generate different training and test examples, the resulting training and test data streams consisted of about 80.000 and 20.000 sensor and motor recordings respectively. With a Khepera communication time step of $\Delta t = 25\,\text{ms}$, this training data stream is equivalent to a real-time experiment with a duration of about 2000 seconds and a test run of about 500 seconds. For a neural microcircuit of this size and connection density the simulation of 2000 seconds takes about 1 minute.

Each single experiment was carried out identically: first, the training data stream was fed into the simulated neural microcircuit via the 6 input neurons. The spike times of all intra-column neurons were recorded. This – in this case 54 dimensional – vector of spiking times, folded with the model of the postsynaptic current $e^{-\frac{t}{\tau_s}}$ is referred to the state vector $x(t)$ of the liquid state machine. The two readouts

of the neural microcircuit model are trained with linear regression to output the target wheel speeds of the left and right motors as imposed by the Braitenberg type controller. After training, the calculated weights w_L and w_R are set for each synapse and the test data stream is connected to the input neurons.

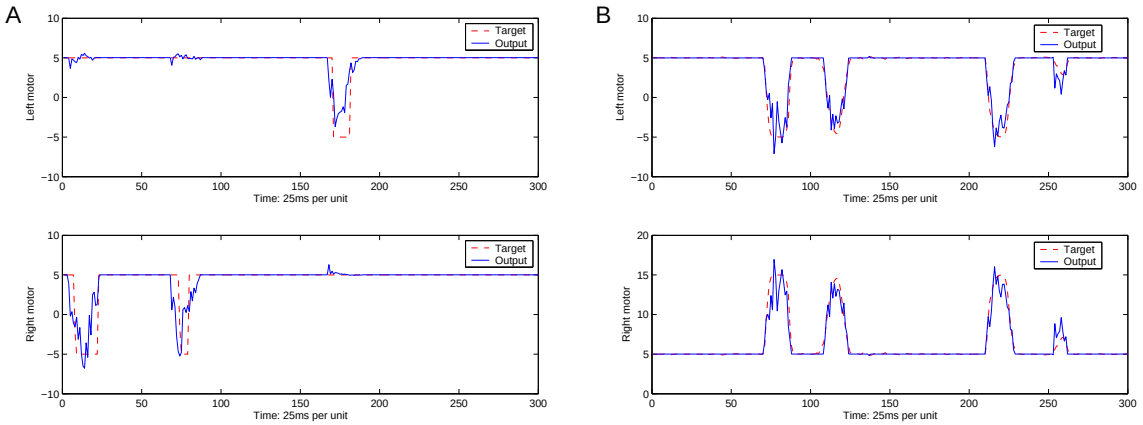

Figure 4: *Snapshots of 7.5 seconds of experiments with two controller types, illustrating the motor control values of the behavioral template (dotted lines) compared to the output of the readout neurons (solid lines) during the testing phase after training. Figures A and B show results for Braitenberg and non-linear type controller respectively*

The motor commands generated during the test phase are recorded and compared to the target values of the Braitenberg type controller. For a session of 50 experiments, the average correlation coefficient between the recorded and the target vectors are 0.9334 for the left motor and 0.9338 for the right motor. A snapshot of one of the experiments is shown in Figure 4. Since the motor of the Khepera acts as a low-pass filter the actual robot movements during the real-time runs become smoother than the plots suggest. Experiments with the non-linear controller yielded similar results. In this case the correlation coefficients for the left and right motor are 0.8130 and 0.8263 respectively. Figure 4 illustrates that it is "harder" for the neural microcircuit to imitate a non-linear behavior like the step-functions that are incorporated in this type of controller. Due to the low-pass filter characteristics of a DC-motor, the desired behavior of Khepera is well recognizable.

5 Discussion

We presented a new approach for a spike based robotic learning systems that operates with real-world devices. It revealed, that a single neural microcircuit suffices as the main computational unit for many robotic controlling tasks. [13] show further results of experiments based on this controller type for a simulated 2-joint robot arm. They use a closed loop controller to control the behavior of the arm. In contrast to our open loop setup, this approach yields a more stable controller for many robotic tasks but it is more complicated to setup, since the feedback has to be precisely controlled to satisfy the well-known BIBO-criteria (bounded input, bounded output). Also, this setup has still to prove the stability in real-world experiments with noise and synchronization problems.

The results of this article now also provide a common platform available for experiments involving spiking neural networks and real-world devices. The generic communication and synchronization approach does not limit this platform to robot control. Instead, one can use arbitrary real-world sensors and actuators as input to and output from a spiking neural network. A drawback of the current implementation of the imitation learning experiments is, that learning has to be broken up in many separate parts: a phase where the robot is controlled by e.g. a programmed controller and all sensor and motor values are recorded. Then the weights of the readout neurons are calculated with linear optimization. Next, these weights are

set in the simulated neural microcircuit. Eventually, the robot can be controlled by the simulated spiking neural network. A more intuitive approach to *imitation learning* would be e.g. to be able to control a robot with a programmed controller or even with a joystick and the learning is done *in parallel*. Since linear optimization suffices at least for simple types of robot controllers, available online versions of this algorithm could be incorporated to achieve this kind of imitation learning. Online versions of the Least Mean Squared (LMS) and the Recursive Least Squares (RLS) algorithms provide promising simplicity with arguable computational costs.

The behaviors used as templates for the initial learning experiments in this article were restricted to reactive controllers. These controllers do not fully expose the inherent temporal integration capabilities of the liquid state machine. A liquid state machine with a single neural microcircuit is able to generate more complex behaviors e.g. with delays between sensors readings and motor control output as it is shown in [13] and [1]. Theoretically, the idea of using neural microcircuits is even more extensible. A single neural column can be used to provide a liquid state for many different readouts that can be trained to expose different controls. Examples for several different readout functions from a single neural column are given in [8]. Furthermore, one could use many similar neural microcircuits, each trained to output certain functions in time. The outputs can then be used again as input to further neural microcircuits. The organization of mammal brains suggest that such hierarchies of neural microcircuits are used to generate complex control behaviors. The framework developed during the work for this article provides a generic basis for future experiments testing such hypothesis.

References

[1] H. Burgsteiner, M. Kröll, A. Leopold, and G. Steinbauer. Movement prediction from real-world images using a liquid state machine. In *Proceedings of the 18th International Conference IEA/AIE*, Lecture Notes in Artificial Intelligence. Springer, to appear 2005.

[2] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:396–407, 1996.

[3] H. Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical Report 148, GMD, 2001.

[4] F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

[5] B.A. Pearlmutter. Gradient calculation for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995.

[6] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press, 1995.

[7] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[8] W. Maass, T. Natschlaeger, and T. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[9] A. Gupta, Y. Wang, and H. Markram. Organizing principles for a diversity of gabaergic interneurons and synapses in the neocortex. *Science*, 287:273–278, 2000.

[10] A.M. Thomson, D.C. West, Y. Wang, and A.P. Bannister. Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2-5 of adult rat and cat neocortex: Triple intracellular recordings and biocytin labelling in vitro. *Cerebral Cortex*, 12(9):936–953, 2002.

[11] H. Markram, Y. Wang, and M. Tsodyks. Differential signaling via the same axon of neocortical pyramidal neurons. *PNAS*, 95(9):5323–5328, 1998.

[12] T. Natschläger, H. Markram, and W. Maass. Computer models and analysis tools for neural microcircuits. In R. Kötter, editor, *Neuroscience Databases. A Practical Guide*, pages 123–138, Boston, 2003. Kluwer Academic Publishers.

[13] P. Joshi and W. Maass. Movement generation and control with generic neural microcircuits. In *Proceedings of BIO-ADIT*, 2004.

[14] H. Burgsteiner. Imitation learning with spiking neural networks and real-world devices. *in submission*, 2005.