Movement Prediction from Real-World Images Using a Liquid State Machine*

Harald Burgsteiner[1], Mark Kröll[2], Alexander Leopold[2], and Gerald Steinbauer[3]

[1] InfoMed/Health Care Engineering, Graz University of Applied Sciences, Eggenberger Allee 9-11, A-8020 Graz, Austria
[2] Institute for Theoretical Computer Science, Graz University of Technology, Inffeldgasse 16b/I, A-8010 Graz, Austria
[3] Institute for Software Technology, Graz University of Technology, Inffeldgasse 16b/II, A-8010 Graz, Austria

Abstract. Prediction is an important task in robot motor control where it is used to gain feedback for a controller. With such a self-generated feedback, which is available before sensor readings from an environment can be processed, a controller can be stabilized and thus the performance of a moving robot in a real-world environment is improved. So far, only experiments with artificially generated data have shown good results. In a sequence of experiments we evaluate whether a liquid state machine in combination with a supervised learning algorithm can be used to predict ball trajectories with input data coming from a video camera mounted on a robot participating in the RoboCup. This pre-processed video data is fed into a recurrent spiking neural network. Connections to some output neurons are trained by linear regression to predict the position of a ball in various time steps ahead. Our results support the idea that learning with a liquid state machine can be applied not only to designed data but also to real, noisy data.

1 Introduction

The prediction of time series is an important issue in many different domains, such as finance, economy, object tracking, state estimation and robotics. The aim of such predictions could be to estimate the stock exchange price for the next day or the position of an object in the next camera frame based on current and past observations. In the domain of robot control such predictions are used to stabilize a robot controller. See [1] for a survey of different approaches in motor control where prediction enhances the stability of a controller. A popular approach is to learn the prediction from previously collected data. The advantages are that knowledge of the internal structure is not necessarily needed, arbitrary non-linear prediction could be learned and additionally some past observations could be integrated in the prediction.

* Authors are listed in alphabetical order.

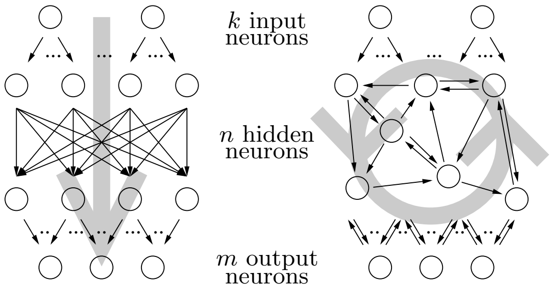

Fig. 1. Comparison of the architecture of a feed-forward (left hand side) with a recurrent neural network (right hand side); the grey arrows sketch the direction of computation

Artificial Neural Networks (ANN) are a common method used for this computation. *Feed-forward networks* only have connections starting from external input nodes, possibly via one or more intermediate hidden node processing layers, to output nodes. *Recurrent networks* may have connections feeding back to earlier layers or may have lateral connections (i.e. to neighboring neurons on the same layer). See Figure 1 for a comparison of the direction of computation between a feed-forward and a recurrent neural network. With this recurrency, activity can be retained by the network over time. This provides a sort of memory within the network, enabling it to compute functions that are more complex than just simple reactive input-output mappings. This is a very important feature for networks that will be used for computation of time series, because a current output is not solely a function of the current sensory input, but a function of the current and previous sensory inputs and also of the current and previous internal network states. This allows a system to incorporate a much richer range of dynamic behaviors. Many approaches have been elaborated on recurrent ANNs. Some of them are: dynamic recurrent neural networks, radial basis function networks, Elman networks, self-organizing maps, Hopfield nets and the "echo state" approach from [2].

Recently, networks with models of biologically more realistic neurons, e.g., spiking neurons, in combination with simple learning algorithms have been proposed as general powerful tools for the computation on time series [3]. In Maass et. al. [4] this new computation paradigm, a so called *Liquid State Machine* (LSM), was used to predict the motion of objects in visual inputs. The visual input was presented to a 8x8 sensor array and the prediction of the activation of these sensors representing the position of objects for succeeding time steps was learned. This approach appears promising, as the computation of such prediction tasks is assumed to be similar in the human brain [5]. The weakness of the experiments in [4] is that they were only conducted on artificially generated data. The question is how the approach performs with real-world data. Real data, e.g. the detected motion of an object in a video stream from a camera mounted on a moving robot, are noisy and afflicted with outliers.

In this paper we present how this approach can be extended to a real world task. We applied the proposed approach to the RoboCup robotic-soccer domain. The task was movement prediction for a ball in the video stream of the robot's camera. Such a prediction is important for reliable tracking of the ball and for decision making during a game. The remainder of this paper is organized as follows. The next section provides an overview of the LSM. Section 3 describes the prediction approach for real data. Experimental results will be reported in Section 4. Finally, in Section 5 we draw some conclusions.

2 The Liquid State Machine

2.1 The Framework of a Liquid State Machine

The "liquid state machine" (LSM) from [3] is a new framework for computations in neural microcircuits. 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* that 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.

Recurrent neural networks with spiking neurons represent a non-linear dynamical system with a high-dimensional internal state, which is driven by the input. The internal state vector $x(t)$ is given as the contributions of all neurons within the LSM to the membrane potential of a readout neuron at the time t. The complete internal state is determined by the current input and all past inputs that the network has seen so far. Hence, a history of (recent) inputs is preserved in such a network and can be used for computation of the current output. The basic idea behind solving tasks with a LSM 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. only have to minimize the mean square error in relation to a desired output can be applied.

The LSM has several interesting features in comparison to other approaches with recurrent circuits of spiking neural networks:

1. The liquid state machine provides "any-time" computing, i.e. one does not have to wait for a computation to finish before the result is available. Results start emitting from the readout neurons as soon as input is fed into the liquid. Furthermore, different computations can overlap in time. That is, new input can be fed into the liquid and perturb it while the readout still gives answers to past input streams.
2. A single neural microcircuit can not only be used to compute a special output function via the readout neurons. Because the LSM only serves as a pool for dynamic recurrent computation, one can use many different readout neurons to extract information for several tasks in parallel. So a sort of "multi-tasking" can be incorporated.

3. In most cases simple learning algorithms can be used to set the weights of the readout neurons. The idea is similar to support vector machines, where one uses a kernel to project input data into a high-dimensional space. In this very high-dimensional space simpler classifiers can be used to separate the data than in the original input data space. The LSM has a similar effect as a kernel: due to the recurrency the input data is also projected to a high-dimensional space. Hence, in almost any case experienced so far simple learning rules like e.g. linear regression suffice.
4. Last but not least it is not only a computational powerful model, but it is also one of the biological most plausible so far. Thus, it provides a hypothesis for computation in biological neural systems.

The model of a neural microcircuit as it is used in the LSM is based on evidence found in [6] and [7]. Still, it gives only a rough approximation to a real neural microcircuit since many parameters are still unknown. The neural microcircuit is the biggest computational element within the LSM, although multiple neural microcircuits could be placed within a single 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. One also specifies a factor to determine how many of the N neurons should be inhibitory. Another important parameter in the definition of a neural microcircuit is the parameter λ. Number and range of the connections between the N neurons within the LSM are determined by this parameter λ. The probability of a connection between two neurons i and j is given by $p_{(i,j)} = C \cdot exp^{-\frac{D_{(i,j)}}{\lambda^2}}$ where $D_{(i,j)}$ is the Euclidean distance between those two neurons and C is a parameter depending on the type (excitatory or inhibitory) of each of the two connecting neurons. There exist 4 possible values for C for each connection within a neural microcircuit: C_{EE}, C_{EI},C_{IE} and C_{II} may be used depending on whether the neurons i and j are excitatory (E) or inhibitory (I). 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 $v(t) = w \cdot e^{-\frac{t}{\tau_{syn}}}$ where w is a synaptic weight and τ_{syn} is the synaptic time constant. In case of dynamic synapses the "weight" w depends on the history of the spikes it has seen so far according to the model from [8]. For synapses transmitting analog values (such as the output neurons in our experimental setup) synapses are simply modeled as static synapses with a strength defined by a constant weight w. Additionally, synapses for analog values can have delay lines, modeling the time a potential would need to propagate along an axon.

3 Experimental Setup

In this section we introduce the general setup that was used during our experiments to solve prediction tasks with real-world data from a robot. As depicted in figure 2, such a network consists of three different neuron pools: (a) an input layer that is used to feed sensor data from the robot into the network, (b) a pool of neurons forming the LSM according to section 2 and (c) the output layer consisting of readout neurons which perform a linear combination of the membrane potentials obtained from the liquid neurons.

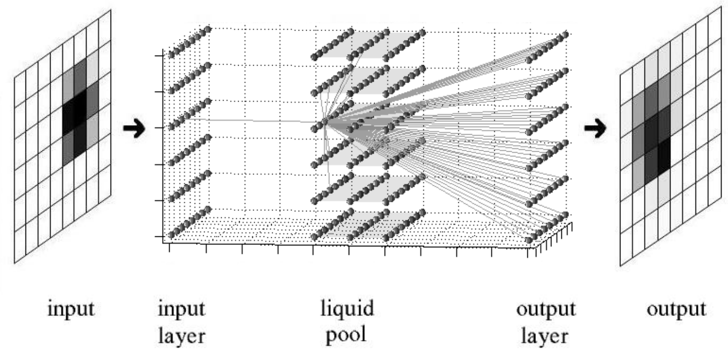

Fig. 2. Architecture of our experimental setup depicting the three different pools of neurons and a sample input pattern with the data path overview. Example connections of a single liquid neuron are shown: input is received from the input sensor field on the left hand side and some random connection within the liquid. The output of every liquid neuron is projected onto every output neuron (located on the most right hand side). The 8x6x3 neurons in the middle form the "liquid"

For simulation within the training and evaluation the neural circuit simulator *CSim*[1] was used. Parameterization of the LSM is described below. Names for neuron and synapse types all originate from terms used in the *CSim* environment. Letters I and E denote values for inhibitory and excitatory neurons respectively.

To feed activation sequences into the liquid pool, we use *External Input Neurons* that conduct an injection current I_{inject} via *Static Analog Synapses* (I_{noise} = 0nA, $w_{mean} = 3 * 10^{-8}$ (EE) or $6 * 10^{-8}$ (EI), $delay_{mean}$ = 1.5ms (EE) or 0.8ms (EI) with CV =0.1) into the first layer of the liquid pool. EE, EI, IE and II denote connections between the two types of neurons. Inspired from information processing in living organisms, we set up a cognitive mapping from input layer to liquid pool. The value of I_{inject} depends on the value of the input data, in this case the activation of each single visual sensor.

The liquid consists of *Leaky Integrate And Fire Neurons* (C_m = 30nF, R_m = 1$M\Omega$, V_{thresh} = 15mV, $V_{resting}$ = 0mV, V_{reset} uniform distributed in the interval [13.8mV 14.5mV], V_{init} uniform distributed in the interval [13.5mV 14.9mV], $T_{refract}$ = 3ms (E) or 2ms (I), I_{noise} = 0nA, I_{inject} uniform distributed in the

[1] The software simulator *CSim* and the appropriate documentation for the liquid state machine can be found on the web page `http://www.lsm.tugraz.at/`

interval [13.5nA 14.5nA]), grouped in an $8 \cdot 6 \cdot 3$ cuboid, that are randomly connected via *Dynamic Spiking Synapses* ($U_{mean} = 0.5, 0.05, 0.25, 0.32$, $D_{mean} = 1.1, 0.125, 0.7, 0.144$; $F_{mean} = 0.05$s, 1.2s, 0.02s, 0.06s; $delay_{mean} = 1.5$ms, 0.8ms, 0.8ms, 0.8ms with $CV = 0.1$; $\tau_{syn} = 3$ms, 3ms, 6ms, 6ms; for EE, IE, EI, II), as described above. The probability of a connection between every two neurons is modeled by the probability distribution depending on a parameter λ described in the previous section. Various combinations of λ (connection probability) and mean connection weights Ω (connection strength) were used for simulation. 20% of the liquid neurons were randomly chosen to produce inhibitory potentials. C was chosen to be 0.3 (EE), 0.4 (EI), 0.2 (IE) and 0.1 (II). Figure 2 shows an example for connection within the LSM.

The information provided by the spiking neurons in the liquid pool is processed (read out) by *External Output Neurons* (V_{init}, $V_{resting}$, I_{noise} are the same as for the liquid neurons), each of them connected to all neurons in the liquid pool via *Static Spiking Synapses* ($\tau_{syn} = 3$ms (EE) or 6ms (EI), $w = -6.73 * 10^{-5}$ (e.g., set after training), $delay_{mean} = 1.5$ms (EE) or 0.8ms (EI) with $CV = 0.1$). The output neurons perform a simple linear combination of inputs that are provided by the liquid pool.

We evaluate the prediction approach by carrying out several experiments with real-world data in the RoboCup Middle-Size robotic soccer scenario. The experiments were conducted using a robot of the "Mostly Harmless" RoboCup Middle-Size team [9]. The task within the experiments is to predict the movement of the ball in the field of view a few frames into the future. The experimental setup can be described as follows: The robot is located on the field and points its camera across the field. The camera is a color camera with a resolution of 320 times 240 pixel. The ball is detected within an image by simple color-blob-detection leading to a binary image of the ball. We can use this simple image preprocessing since all objects on the RoboCup-field are color-coded and the ball is the only red one. The segmented image is presented to the 8 times 6 sensor field of the LSM. The activation of each sensor is equivalent to the percentage of how much of the sensory area is covered by the ball.

We collect a large set of 674 video sequences of the ball rolling with different velocities and directions across the field. The video sequences have different lengths and contain images in 50ms time steps. These video sequences are transfered into the equivalent sequences of activation patterns of the input sensors. Figure 3 shows such a sequence. The activation sequences are randomly divided into a training set (85%) and a validation set (15%) used to train and evaluate the prediction. Training and evaluation is conducted for the prediction of 2 timesteps (100ms), 4 timesteps (200ms) and 6 timesteps (300ms) ahead. The corresponding target activation sequences are simply obtained by shifting the input activation sequences 2, 4 or 6 steps forward in time.

Simulation for the training set is carried out sequence-by-sequence: for each collected activation sequence, the neural circuit is reset, input data are assigned to the input layer, recorders are set up to record the liquid's activity, simulation is started, and the corresponding recorded liquid activity is stored for the

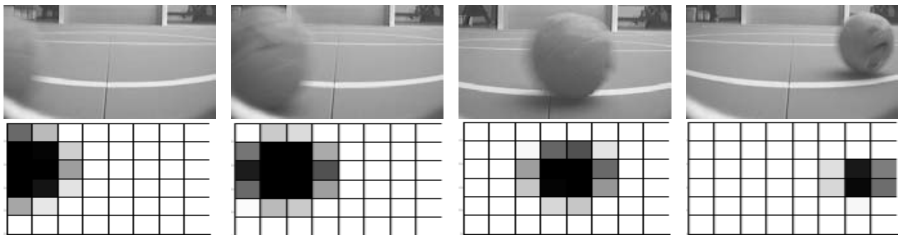

Fig. 3. Upper Row: Ball movement recorded by the camera. Lower Row: Activation of the sensor field

training part. The training is performed by calculating the weights[2] of all static synapses connecting each liquid neuron with all output layer neurons using linear regression.

Analogous to the simulation with the training set, simulation is then carried out on the validation set of activation sequences. The resulting output neuron activation sequences (*output sequences*) are stored for evaluating the network's performance.

4 Results

We introduce the mean absolute error and the correlation coefficient to evaluate the performance of the network. The mean absolute error is the positive difference between the activation values of target and output sequences of the validation set divided by the number of neurons in the input/output layer and the length of the sequence. This average error per output neuron and per image yields a reasonable measure for the performance on validation sets with different length. Figure 4 shows an example for a prediction and its error.

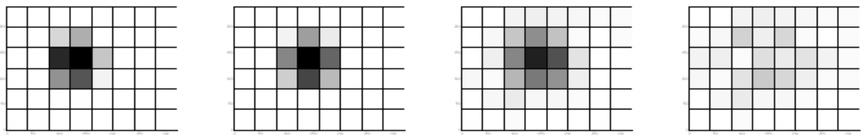

Fig. 4. Sensor activation for a prediction one timestep ahead. Input activation, target activation, predicted activation and error (left to right)

A problem which arises if only the mean absolute error is used for evaluation is that also networks with nearly no output activation produce a low mean

[2] In fact also the injection currents I_{inject} for each output layer neuron is calculated. For simplification this bias is treated as the 0^{th} weight

absolute error - because most of the neurons in the target activation pattern are not covered by the ball and therefore they are not activated leading to a low average error per image. The correlation coefficient measures the *linear* dependency of two variables. If the value is zero two variables are not correlated. The correlation coefficient is calculated in similar way as the mean absolute error. Therefore the higher the coefficient the higher the probability of getting a correlation as large as the observed value without coincidence involved. In our case a relation between mean absolute error and correlation coefficient exists. A high correlation coefficient indicates a low mean absolute error.

In Figure 5 the mean absolute errors averaged over all single images in the movies in the validation set and the correlation coefficients for the prediction one timestep (50ms) ahead are shown for various parameter combinations. The parameter values range for both landscapes from 0.1 to 5.7 for Ω and from 0.5 to 5.7 for λ. If both Ω and λ are high, there is too much activation in the liquid. Remember, λ controls the probability of a connection and Ω controls the strength of a connection. We assume that this high activity hampers the network making a difference between the input and the noise. Both values indicate a good area if at least one of the parameters is low. Best results are achieved if both parameters are low (e.g. Ω=0.5, λ=1.0). The figure clearly shows the close relation between the mean absolute error and the correlation coefficient. Furthermore, it shows the very good results for the prediction as the correlation coefficient is close to 1.0 for good parameter combinations.

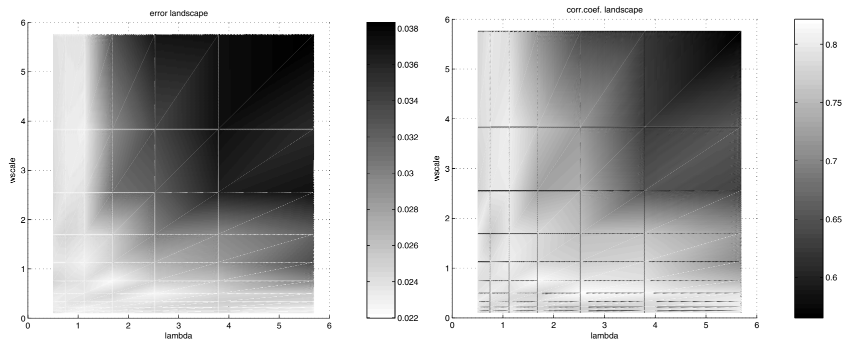

Fig. 5. Mean absolute error landscape on the left and correlation coefficient on the right for a prediction one time step ahead. $\Omega(wscale)$ [0.1,5.7], λ [0.5,5.7]

We also compare the results achieved with two (100ms) and four (200ms) time steps predicted. In order to compare the results of both predictions for different parameter combinations, we use again a landscape plot of the correlation coefficients. Figure 6 shows the correlation coefficient for parameter values range from 0.1 to 5.7 for Ω and from 0.5 to 5.7 for λ. The regions of good results remain the same as in the one timestep prediction. If at least one parameter - Ω

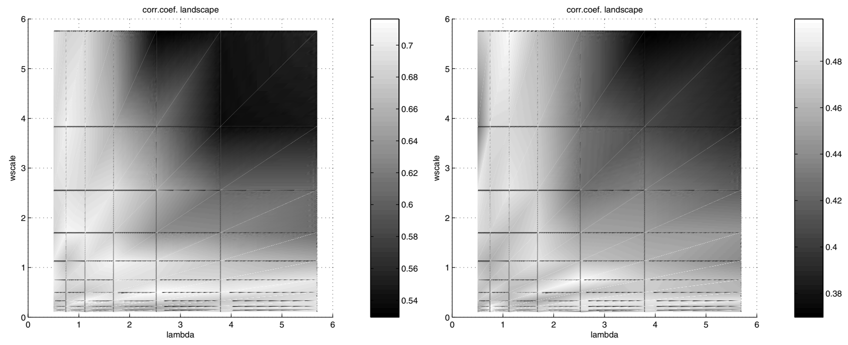

Fig. 6. Correlation coefficient landscape for two timesteps (100ms) on the left hand side and four timesteps (200ms) on the right hand side

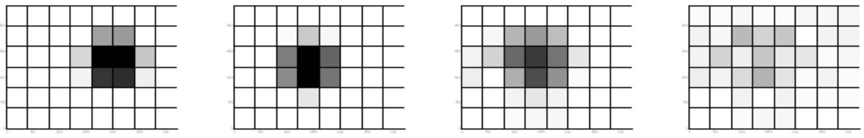

Fig. 7. Sensor activation for a prediction two timesteps ahead. Input activation, target activation, predicted activation and error (left to right). Parameter: Ω=1.0, λ=2.0

or λ - is low the correlation coefficient reaches its maximum (about 0.7 at two timesteps and about 0.5 at four timesteps). With increasing Ω and λ, the correlation coefficients decrease again. We believe that the too high activation is again the reason for this fact. Not surprisingly the maximum correlation compared to the one step prediction is lower because prediction gets harder if the prediction time increases. Nevertheless, the results are good enough for reasonable predictions.

Figure 7 shows an example for the activations and the error for the prediction of two timesteps ahead. It clearly shows that the center of the output activation is in the region of high activation in the input and the prediction is reasonable good. The comparison to Figure 4 also shows that the activation is more and more blurred around its center if the prediction time increases.

Furthermore we confronted the liquid with the task to predict 300ms (6 timesteps) without getting a proper result. We were not able to visually identify the ball position anymore. We guess this is mainly caused by the blur of the activation.

5 Conclusion and Future Work

In this work we propose a biologically more realistic approach for the computation of time series of real world images. The *Liquid State Machine (LSM)*, a new biologically inspired computation paradigm, is used to learn ball prediction within the RoboCup robotic soccer domain. The advantages of the LSM are that it projects the input data in a high-dimensional space and therefore simple learning methods, e.g. linear regression, can be used to train the readout. Furthermore, the *liquid*, a pool of inter-connected neurons, serves as a memory which holds the current and some past inputs up to a certain point in time (fading memory). Finally, this kind of computation is also biologically more plausible than other approaches like Artificial Neural Networks or Kalman Filters. Preliminary experiments within the RoboCup domain show that the LSM approach is able to reliably predict ball movement up to 200ms ahead. But there are still open questions. One question is how the computation is influenced by the size and topology of the LSM. Moreover, deeper investigation should be done for more complex non-linear movements, like balls bouncing back from an obstacle. Furthermore, it might be interesting to directly control actuators with the output of the LSM. We currently work on a goalkeeper, which intercepts the ball, controlled directly by the LSM approach.

References

1. M.I. Jordan and D.M. Wolpert. Computational motor control. In M. Gazzaniga, editor, *The Cognitive Neurosciences.* MIT Press, Cambridge, MA, 1999.
2. H. Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical Report 148, GMD, 2001.
3. 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.
4. W. Maass, R. A. Legenstein, and H. Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. In H. H. Buelthoff, S. W. Lee, T. A. Poggio, and C. Wallraven, editors, *Biologically Motivated Computer Vision. Proc. of the Second International Workshop, BMCV 2002*, volume 2525 of *Lecture Notes in Computer Science*, pages 282–293. Springer (Berlin), 2002.
5. M. F. Bear. *Neuroscience: Exploring the brain.* Williams and Wilkins, Baltimore, MA, 2000.
6. 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.
7. 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.
8. H. Markram, Y. Wang, and M. Tsodyks. Differential signaling via the same axon of neocortical pyramidal neurons. *PNAS*, 95(9):5323–5328, 1998.
9. G. Fraser, G. Steinbauer, and F. Wotawa. A modular architecture for a multi-purpose mobile robot. In *Innovations in Applied Artificial Intelligence, IEA/AIE*, volume 3029 of *Lecture Notes in Artificial Intelligence*, Canada, 2004. Springer.