Simulation of Spiking Neural Networks

Olaf Booij
obooij@science.uva.nl

19th May 2004

Abstract

This brief technical writing will explain an efficient way of simulating spiking neural networks (SNNs). Because SNNs consist of a network of non-linear dynamic computational units, simulation on a serial computer uses a lot of resources. It is therefor important for the research on and application of SNNs that there is an efficient way to implement these algorithms. We will describe a way to efficiently simulate a generic spiking neuron model, called the Spike Response Model, so that it can be used for large scale networks.

1 Introduction

In the last ten years there has been a shift in neural network research from the conventional rate-coded neural networks to the more biological plausible spike-coded neural networks [15][1][7]. The implementation of these spiking neural networks (SNNs) is somewhat more involved then conventional networks, because spiking neurons are, as opposed to the static sigmoidal neurons, dynamic systems.

Specific SNN-models, like the Spike Response Model, usually consist of a set of formulas that describe the state of the spiking neurons at a certain time-step given all the input the neuron received until then. This is convenient if we want to make a mathematical analyses of SNNs, but is causes trouble when we want to simulate them. The problem is that we need to remember all the spikes of all the neurons that happened in the past, which can cost a lot of memory. But much more troublesome is that we need to calculate the effect of these spikes on the neuron's state for every time-step, taking up a lot of computation-time.

To counter this problem a lot of proposals have been done to transform the state-formulas of spiking neurons to recursive functions that express the state of a neuron at time t in the previous state at time $t - \Delta t$ [9][4][5][16]. But all these systems restrict the neuron models in such a way that it limits the spiking neuron's power. To be more specific: the form of the potential change due to a presynaptic spike is modeled by a simple exponential decay, but research has pointed out that the computational power of a spiking neuron is due to the initial rising of the potential in response to an incoming spike [12][2][14][13]. In this writing we will show that SNNs with more accurate neuron models can also be simulated with efficient recursive functions. We will first describe the specific spiking neuron model we will use in section 2. And then in section 3 we will explain how this model can be simulated efficiently.

2 Spike Response Model

In this section we will present the spiking neuron model we will use, called the Spike Response Model (SRM) [6]. The SRM formally describes how the incoming spike-trains are processed to produce a new spike-train leaving the neuron.

The state of spiking neuron j in the SRM is described by its potential $u_j(t)$. When this potential crosses a certain threshold-value ϑ the neuron fires a spike, that is describes by its spike-time $t^{(f)}$.

The output of neuron j is thus fully characterized by the array of spike-times:

$$\mathcal{F}_j = \{t_j^{(f)}; 1 \leq f \leq n\} = \{t | u_j(t) = \vartheta\}, \tag{1}$$

where n denotes the number of spikes. The spike-train $\mathcal{F}_j$ is chronologically ordered so if $1 \leq f < g \leq n$, then $t_j^{(f)} < t_j^{(g)}$.

The potential of a neuron can change due to spikes of its presynaptic neurons $i \in \Gamma_j$, where

$$\Gamma_j = \{i | i \text{ is presynaptic to } j\}. \tag{2}$$

If presynaptic neuron i has fired a spike at time $t_i^{(g)} \in \mathcal{F}_i$ the potential of postsynaptic neuron j at time t is raised by $w_{ji}\varepsilon(t - t_i^{(g)} - d_{ji})$. The variable w_{ji} denotes the weight of the connection and d_{ji} denotes the delay of the connection. The spike response function ε describes the effect the presynaptic spike has on the potential of the postsynaptic neuron. Different mathematical formulations are possible but the shape of the function is always a short rising part followed by a long decaying part and $\varepsilon(s) = 0$ for $s \leq 0$ to insure causality, see figure 1.

The formula we will use is the difference of two exponential decays [6]:

$$\varepsilon(s) = \left[\exp\left(-\frac{s}{\tau_m}\right) - \exp\left(-\frac{s}{\tau_s}\right)\right] \mathcal{H}(s), \tag{3}$$

where $\mathcal{H}(s)$ denotes the Heavy-side step function: $\mathcal{H}(s) = 0$ for $s \leq 0$ and $\mathcal{H}(s) = 1$ for $s > 0$. The two time-constants τ_m and τ_s (with $0 < \tau_s < \tau_m$) control the steepness of the rise and the decay of the function and more importantly they control the position of the top of the function.

Another process that changes the potential of a neuron is the refractoriness. This is modeled with the refractory function η. If neuron j emitted a spike at $t_j^{(f)}$ its potential at t is lowered with $\eta(t - t_j^{(f)})$. Again the Simple Response Model does not fix this function, but to insure causality it is required that

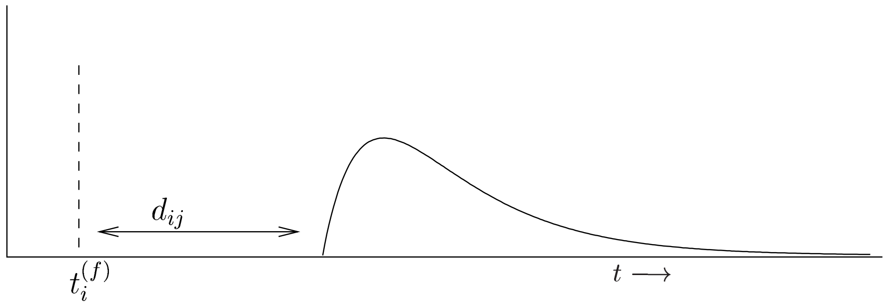

Figure 1: The change in the potential of postsynaptic neuron j, given a spike of presynaptic neuron i is given by $\varepsilon(t - t_i^{(f)} - d_{ji})$. At time $t_i^{(f)}$ the presynaptic neuron i fires a spike, shown in the figure with the dashed vertical line. After the delay of the connection, d_{ji}, the spike has an effect on neuron j. First the potential of postsynaptic neuron j increases fast, followed by a long decay until the spike has no influence anymore.

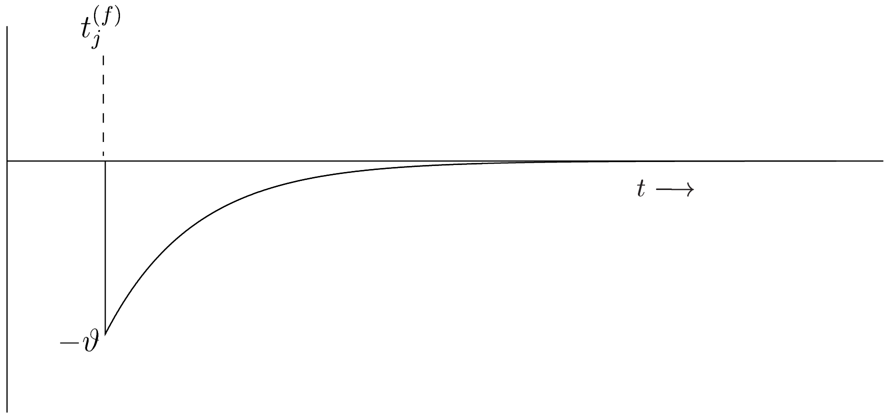

Figure 2: The change in the potential of neuron j given a spike is given by $\eta(t - t_j^{(f)})$. At time $t_j^{(f)}$ neuron j fires a spike, shown in the figure with the dashed vertical line. Immediately after the spike the potential of postsynaptic neuron j drops to $-\vartheta$ and then slowly recovers, followed by a long decay until the spike has no influence anymore.

$\eta(s) = 0$ for $s \leq 0$ and it is usually non-positive, see figure 2. We use a simple exponential decay:

$$\eta(s) = -\vartheta \exp\left(-\frac{s}{\tau_r}\right) \mathcal{H}(s), \tag{4}$$

where ϑ is the threshold of the neuron, $\mathcal{H}(s)$ is the Heavy-side step function and τ_r is another time-constant. Usually this time-constant is big enough so that refractoriness lasts longer then the spike response function: $\forall s \eta(s) > -\varepsilon(s)$.

The formula of the potential of a neuron, $u_j(t)$, is the sum of the influence of the presynaptic spikes and of its own spikes:

$$u_j(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta(t - t_j^{(f)}) + \sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} w_{ji} \varepsilon(t - t_i^{(g)} - d_{ji}) \tag{5}$$

Equations 1 and 5 define the Spike Response Model and together with equations 3 and 4 they fully describe the behavior of one single neuron. A simple example illustrating the process is shown in figure 3, where the spikes of two neurons cause a third neuron two fire.

3 Simulation of Spike Response Model

This section describes how to efficiently simulate spiking neural networks.

The Spike Response Model (SRM) is a set of continuous formulas, that tells us if a spiking neuron fires a spike at time t given certain input. To compute all the firing times $t^{(f)} \in \mathcal{F}$ of a neuron we should compute these formulas for all times t in a given time interval. Off course we can only do this for a finite set of moments, so SNNs can only be simulated in discrete-time. Due to this discretization there is an error between the model and the simulation regarding the exact firing time. But if we choose the time steps small enough this error will get smaller then the noise that is present in the biological neurons we want to simulate.

The calculation of the potential of each neuron for each time-step is the most computationally intensive part of the simulation. So the performance of the simulation highly depends on an efficient algorithm, that can compute the equation for the potential (equation (5)):

$$u_j(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta(t - t_j^{(f)}) + \sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} w_{ji} \varepsilon(t - t_i^{(g)} - d_{ji}), \tag{6}$$

By choosing an ε-function and an η-function that can be rewritten in a discrete-time recursive function, we can save a lot of computation time [9].

We will first convert the first part, the refractory-term, of (6), which will be denoted as $u_j^{(R)}$. Filling in the η-function (4) into the first part, we get

$$u_j^{(R)}(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} -\vartheta \exp\left(-\frac{t - t_j^{(f)}}{\tau_r}\right) \mathcal{H}(t - t_j^{(f)}). \tag{7}$$

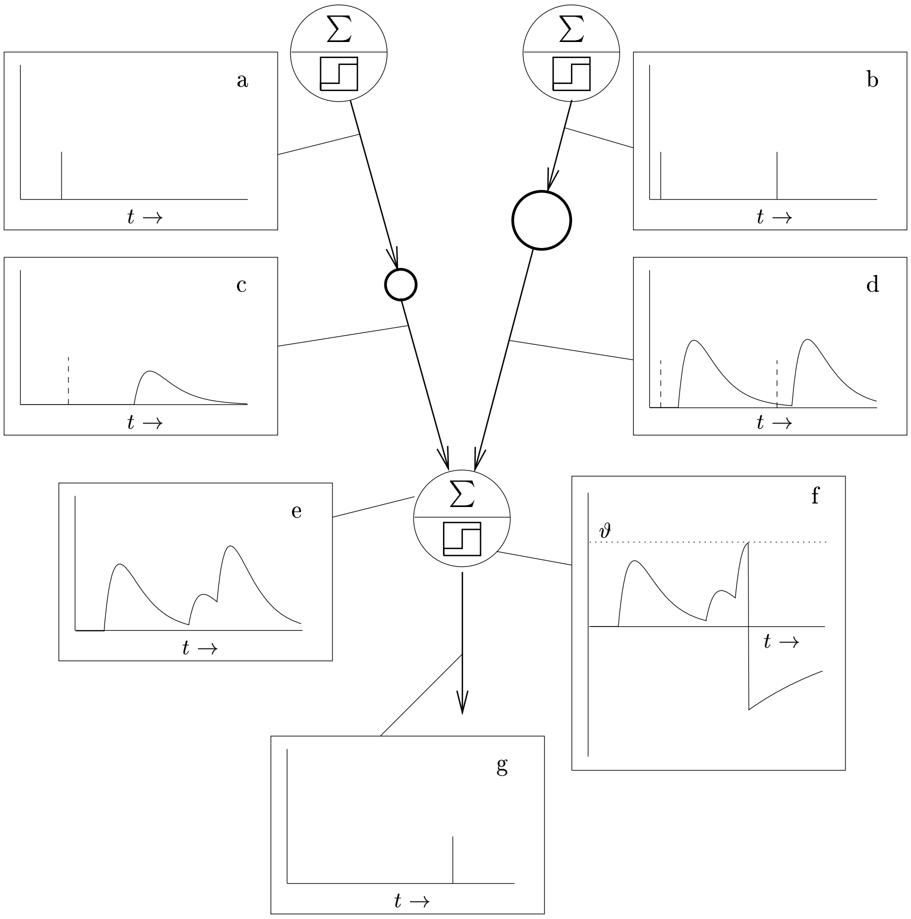

Figure 3: A graphical example explaining the workings of a spiking neuron. a, b: the two top neurons fire spikes, the left fires once, the right fires two times. c, d: the synapses delay and weigh the spikes and then transform them into spike responses using the ε-function; the left synapse has a larger delay then the right synapse, but a smaller weight. e: the incoming spike responses are summed to give the neurons potential. f: the potential is thresholded and because the potential crosses the threshold, the potential drops due to the refractoriness. g: the output of the neuron is a spike at the time the potential reached the threshold.

It is easy to see that we can transform a discretized exponential decay into a recursive function:

$$\begin{aligned} \exp(x) &= \exp(x - \Delta x) \exp(\Delta x) && (8) \\ \exp(0) &= 1, && (9) \end{aligned}$$

where Δx denotes the step-size of the discretization.

Using this the formula (7) can also be transformed into a recursive function:

$$\begin{aligned} u_j^{(R)}(t) &= u_j^{(R)}(t - \Delta t) \exp\left(-\frac{\Delta t}{\tau_r}\right) - \vartheta \sum_{t_j^{(f)} \in \mathcal{F}_j} \delta(t - t_j^{(f)}), && (10) \\ u_j^{(R)}(0) &= 0 && (11) \end{aligned}$$

where Δt denotes the time-step of the discretization and $\delta(\cdot)$ denotes the unit-impulse function with $\delta(0) = 1$ and $\delta(s) = 0$ for $s \neq 0$.

By calculating the refractory-term in this way, we do not only save computation time but also memory space, because we can forget the spike-times of the past and only have to remember a state variable $u_j^{(R)}(t)$.

The same thing can be done with the calculation of the potential change caused by incoming spikes, the second term of equation (6), except that we need two dynamic states, instead of one, for the two exponential terms in the spike response function. Filling in the spike response function (3) in the second part of (6), gives:

$$\sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} w_{ji} \left[\exp\left(-\frac{t - t_i^{(g)} - d_{ji}}{\tau_m}\right) - \exp\left(-\frac{t - t_i^{(g)} - d_{ji}}{\tau_s}\right) \right] \mathcal{H}(t - t_i^{(g)} - d_{ji}), \tag{12}$$

This formula can be split into to functions that we will denote with $u_j^{(M)}(t)$ and $u_j^{(S)}(t)$:

$$\begin{aligned} u_j^{(M)}(t) &= \sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} w_{ji} \exp\left(-\frac{t - t_i^{(g)} - d_{ji}}{\tau_m}\right) \mathcal{H}(t - t_i^{(g)} - d_{ji}), && (13) \\ u_j^{(S)}(t) &= \sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} -w_{ji} \exp\left(-\frac{t - t_i^{(g)} - d_{ji}}{\tau_s}\right) \mathcal{H}(t - t_i^{(g)} - d_{ji}). && (14) \end{aligned}$$

These functions can again be transformed to recursive functions, as with the refractory-term:

$$u_j^{(M)}(t) = u_j^{(M)}(t - \Delta t) \exp\left(-\frac{\Delta t}{\tau_m}\right)$$

$$
\begin{aligned}
& & + \sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} w_{ji} \delta(t - t_i^{(f)} - d_{ji}), & \qquad (15) \\
u_j^{(S)}(t) &=& u_j^{(S)}(t - \Delta t) \exp\left(-\frac{\Delta t}{\tau_s}\right) & \\
& & - \sum_{i \in \Gamma_j} \sum_{t_i^{(g)} \in \mathcal{F}_i} w_{ji} \delta(t - t_i^{(f)} - d_{ji}), & \qquad (16) \\
u_j^{(M)}(0) &=& u_j^{(S)}(0) = 0. & \qquad (17)
\end{aligned}
$$

Again this transformation saves a lot of computation time and memory space. Instead of calculating the response function for every synapse of neuron, only two state variables have to be changed every time-step by a simple multiplication with a constant. The spike-times of the presynaptic neurons need to be stored only for the duration of the delay d_{ji} and not for the duration of the complete simulation as would be the case when using equation 6.

The potential can now be computed by summing up the three state variables:

$$
u_j(t) = u_j^{(M)}(t) + u_j^{(S)}(t) + u_j^{(R)}(t). \qquad (18)
$$

Algorithm 1 summarizes the procedure of one simulation step. The only mathematical operations that are needed are multiplication and addition (notice that the factors $\exp\left(-\frac{\Delta t}{\tau_*}\right)$ are constants and can be calculated beforehand). Therefor this algorithm can efficiently simulate a spiking neural network.

Algorithm 1 SNN simulation step

$u_j^{(M)} \leftarrow u_j^{(M)} \exp\left(-\frac{\Delta t}{\tau_m}\right)$
$u_j^{(S)} \leftarrow u_j^{(S)} \exp\left(-\frac{\Delta t}{\tau_s}\right)$
$u_j^{(R)} \leftarrow u_j^{(R)} \exp\left(-\frac{\Delta t}{\tau_r}\right)$
for all presynaptic spikes $t_i^{(f)}$ **do**
  if $t_i^{(f)} = t - d_{ij}$ **then**
    $u_j^{(M)} \leftarrow u_j^{(M)} + w_{ji}$
    $u_j^{(S)} \leftarrow u_j^{(S)} - w_{ji}$
  end if
end for
if $(u_j^{(M)} + u_j^{(S)} + u_j^{(R)}) \geq \vartheta$ **then**
  Fire a Spike
  $u_j^{(R)} \leftarrow u_j^{(R)} - \vartheta$
end if
$t \leftarrow t + \Delta t$

4 Restrictions on the SRM

The simulation technique we described in the previous section comes at the price of some restrictions to the Simple Response Model, although less severe then the restrictions that come with the other proposals. In this section we give an account of these limitations.

The proposed algorithm assumes that the synapses of one spiking neuron only vary in weight, w_{ji} and delay d_{ji}. The time constants τ_m and τ_s of the spike-response function (equation 3) are the same for all presynaptic spikes of one spiking neuron. The algorithm could however be extended to allow the synapses of one neuron to be divided in different groups which share the same time constants (also indicated in [9]). For each group of synapses a set of two state variables ($u_j^{(M)}$ and $u_j^{(S)}$) should be processed every time-step. For example, some studies use networks made up out of two explicitly different types of neurons, excitory and inhibitory [11][10]. In these studies the response of the postsynaptic potential to a spike of an inhibitory neuron does not only differ in the direction, i.e. the potential decreases, but also lasts longer than the response to an excitory spike, due to larger time-constants. These networks could be simulated with two pairs of state variables for the presynaptic spikes, one pair for the excitory and one for the inhibitory spikes.

In our model we restrict the SRM by using a specific spike-response function ε and refractory function η (equation 3 and 4 respectively). Most studies that use the SRM use the same function for the refractoriness, but for the spike-response however some use different formulas. A function that is commonly used is the so-called alpha-function (for example in [17] and [2]):

$$\varepsilon_\alpha(s) = \frac{s}{\tau} \exp\left(1 - \frac{s}{\tau_b}\right) \mathcal{H}(s), \tag{19}$$

This function has some nice characteristics that makes it easy to manipulate mathematically. The top of the function always has a height of 1 and is located at $s = \tau_b$. And the derivative is a simple function of the formula itself: $\varepsilon'_\alpha(s) = \left(\frac{1}{\tau_b} - \frac{1}{s}\right) \varepsilon_\alpha(s)$. It turns out that the alpha-function behaves almost the same as the spike-response function we use (see figure 4). If we multiply our ε-function (eq. (3)) with a normalization constant C which is given by:

$$C = \left(\frac{\tau_s}{\tau_m}\right)^{\frac{-\tau_s}{\tau_m - \tau_s}} \left(1 - \frac{\tau_s}{\tau_m}\right)^{-1} \tag{20}$$

And set the time-constants τ_m and τ_s given a certain τ_b as follows:

$$\tau_m = \frac{\tau_b}{\ln(2)} \tag{21}$$

$$\tau_s = \frac{\tau_b}{2\ln(2)} \tag{22}$$

Then the two functions share the same top and differ only slightly, see figure 4.

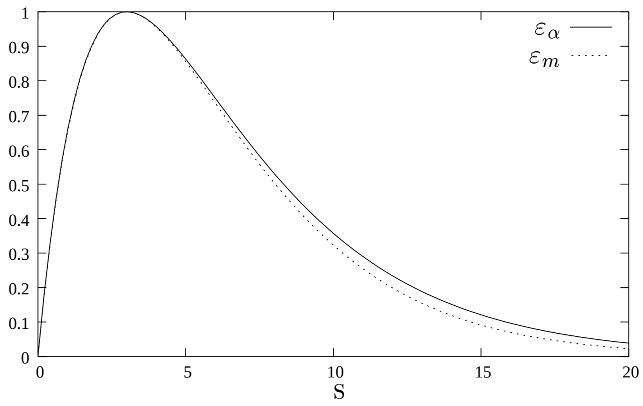

Figure 4: Two ways to model the potential change due to a presynaptic spike. The solid line shows the alpha-function (eq. (19)) with a typical value of 3 for the time-constant τ_b and the dotted line the normalized version of the function we use (eq. (3)). As can be seen, the difference between the two is minimal.

5 Conclusion

The algorithm we designed is easy to implement and use for all kinds of research on large scale spiking neural networks. It uses a biological plausible spike-response function in contrast to other high performance simulation software [18][4][5][16].

This algorithm can be even further optimized by making it event-driven, that is: only updating the state of those neurons that received a spike in the near past [9][18][3][4][5]. In this way the performance of the simulator does not depend on the number time-steps of the simulator but the number of spikes the neurons fire [3][8]. Because spiking neurons are meant to spike only sparsely, this should give a considerable improvement.

Also spiking neural networks are ideal systems to run on parallel computer architectures because there is only little communication between the neurons, especially as compared to conventional sigmoidal neural networks [3][9]. Although it must be noted that making the simulator event-driven, does decrease the effect of running it in parallel.

References

[1] S.M. Bohte. *Spiking Neural Networks.* PhD thesis, Universiteit Leiden, 2003. Available from http://homepages.cwi.nl/˜sbohte/publications2.htm.

[2] S.M. Bohte, J.N. Kok, and H. La Poutré. Error-backpropagation in temporally encoded networks of spiking neurons. Technical report, CWI Technical Report SEN-R0037, 2000. Available from http://db.cwi.nl/rapporten.

[3] C.Wolff C. Grassmann, T. Schnauer. Pcnn neurocomputers - event driven and parallel architectures. In *European Symposium on Artificial Neural Networks ESANN 2002*, pages 331–336, Bruges (Belgium), 2002.

[4] A. Delorme, J. Gautrais, R. VanRullen, and S.J. Thorpe. Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, 26-27:989–996, 1999.

[5] A. Delorme and S. Thorpe. Spikenet: An event-driven simulation package for modeling large networks of spiking neurons. *Network: Computation in Neural Systems*, 14:613–627, 2003.

[6] W. Gerstner. Spiking neurons. In W. Maass and C. M. Bisop, editors, *Pulsed Neural Networks*, pages 3–54. MIT Press (Cambridge), 1999.

[7] W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press, 2002.

[8] A. Jahnke, U. Roth, and H. Klar. "a simd/dataflow architecture for a neurocomputer for spike-processing neural networks (nespinn). In *MicroNeuro'96*, pages 232–237, Lausanne (Switzerland), 1996.

[9] A. Jahnke, U. Roth, and T. Schönauer. Digital simulation of spiking neural networks. In W. Maass and C. M. Bishop, editors, *Pulsed Neural Networks*, pages 237–256. MIT Press (Cambridge), 1999.

[10] A. Knoblauch and G. Palm. Spiking associative memory and scene segmentation by synchronization of cortical activity. In S. Wermter, J. Austin, and D. Willshaw, editors, *Emerging Neural Computational Architectures Based on Neuroscience*, pages 407–427. Springer-Verlag, 2001.

[11] Th. Lindblad, V. Becanovic, C. S. Lindsey, and G. Szekely. Intelligent detectors modelled from the cat's eye. *Nuclear Instruments and Methods in Physics Research Section A*, 389:245–250, feb 1997.

[12] W. Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40, 1996.

[13] W. Maass. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In M. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems, volume 9*, pages 211–217. MIT Press, Cambridge, MA, 1997.

[14] W. Maass. Computing with spiking neurons. In W. Maass and C. M. Bishop, editors, *Pulsed Neural Networks*, pages 55–85. MIT Press (Cambridge), 1999. Available from http://www.cis.tugraz.at/igi/maass/publications.html.

[15] W. Maass and C. M. Bishop, editors. *Pulsed Neural Networks.* MIT Press, Cambridge, 1999.

[16] I. Marian, R. G. Reilly, and D. Mackey. Efficient event-driven simulation for spiking neural networks. In *Proceedings of 3rd WSES International Conference on: Neural Networks and Applications*, Interlaken, Switzerland, 2002.

[17] T. Natschläger and B. Ruf. Spatial and temporal pattern analysis via spiking neurons. *Network: Computation in Neural Systems*, 9(2):319–332, 1998. Available from http://www.igi.tugraz.at/tnatschl/publications.html.

[18] M. Schaefer, T. Schoenauer, C. Wolff, G. Hartmann, H. Klar, and U. Rueckert. Simulation of spiking neural networks - architectures and implementations. *Neurocomputing*, 48(1-4):647–679, 2002.