# RetinotopicNET:

# An Efficient Simulator for Retinotopic Visual Architectures.

Dipl. Eng. RAUL C. MUREŞAN
Nivis Research
Gh. Bilascu, Nr. 85, Cluj-Napoca, Cod 3400
ROMANIA
raulmuresan@personal.ro

*Abstract:* -RetinotopicNET is an efficient simulator for neural networks with retinotopic-like receptive fields. The system has two main characteristics: it is event-driven and it takes advantage of the retinotopic arrangement in the receptive fields. The dynamics of the simulator are driven by the spiking events of the simple integrate-and-fire neurons. By using an implicit synaptic rule to represent the synapses, RetinotopicNET achieves a great reduction of memory requirement for simulation. We show that under such conditions the system is fit for the simulation of very large networks of integrate-and-fire neurons. Furthermore we test RetinotopicNET in the simulation of a complex neural architecture for the ventral visual pathway. We prove that the system is linearly scalable with respect to the number of neurons in the simulation.

*Key-words*: - Neural simulator; Retinotopy; Receptive fields; Event-driven; Spikes.

## 1. Introduction

The evolution of neurocomputing in the last decade led to the necessity of finding strong neural simulators to test researchers' hypotheses. In general the approaches were divided into two main directions. On one hand some tried the creation of problem specific simulators that are limited to the area considered. Such systems have the disadvantage of being too restrictive but their main advantage is that they can be optimized for the very application they were created for. On the other hand some general tools were designed to handle more generic architectures. Still, such general architectures come, sooner or later, to face the same problems as the specific ones. At some point the designer has to get some degree of commitment to ensure that the system can be implemented.

Another recent trend in designing simulators takes into account more complex models of neurons and the fact that the dynamics of the neuron is much more important that its static mathematical model. Spiking neural nets are used more and more frequently, mainly because of their biological plausibility and the relative simplicity of their hardware implementation. Under such circumstances, strong simulators are required to test the architectures before implementing them in hardware.

Recent efforts of researchers led to the creation of efficient neural simulators that can be used for generic architectures. However, they do not attempt to commit the system to specific problems. Such a commitment could yield an increased speed of simulation allowing the system to perform very complex tasks in real-time, on monoprocessor machines. One approach, used in GENESIS [1] allows for the simulation of relatively small networks, of complex, multi-compartment neurons. The system is not fit to simulating large networks, especially when real-time is an issue. Another, completely different approach is fit for very large networks of neurons, used in SpikeNET [2]. SpikeNET is fit for general-purpose simulation for networks of integrate-and-fire neurons. Mattia and Del Giudice [4] propose an algorithm, also for general-purpose simulation, that takes advantage of the event-nature of spikes and also include dynamical synapses into their model.

Our approach tries to unify some general trends in simulators and also to take advantage of problem specific optimizations, such as the retinotopy of receptive fields in visual areas.

## 2. Methods

When trying to build a neural simulator the researcher is faced with two main problems: the amount of memory consumed to represent the

simulation data and the time needed for processing during dynamic updates. We can split these problems into two categories: static problems and dynamic problems.

It can be easily understood that static problems are mainly related to the amount of information considered when representing the neuron's parameters and to the representation of synapses. In general we can consider that the system's complexity is a function of the numbers of neurons in the simulation. In terms of memory consumption, the overload of the system, due to the representation of the neuron's parameters increases linearly. We can give a quantitative expression of this:

$$M_1(n) = k_{neunon} \cdot n \qquad (1)$$

where:

$M_1(n)$ – is the memory consumption induced by n neurons

$k_{neuron}$ – represents the memory consumption induced by one neuron's representation

If adding more memory can easily solve the memory overhead due to the representation of neurons' parameters, the amount of memory required for representing explicitly the synapses can become prohibitive. The number of synapses is a factor of $10^3$ –$10^4$ greater than that of the neurons [4]. When trying to simulate networks with millions of neurons, we are faced with important hardware limitations. Thus, synaptic representation is the main issue in static problems.

On the other hand, dynamic problems impose even greater limitations. There are two aspects that influence the dynamics of a simulator: the complexity of the neuron's model and the design of information transfer during neural updated. In other words, the more complex the neuron, the harder computing is necessary. Then, updating the state of every neuron, in each iteration is a real headache and completely inefficient.

We can find solutions to these problems by imposing some restrictions to the system, as described next.

### 2.1. Solutions to static problems

We saw that the complexity of the neuron and the number of parameters describing it influence both static and dynamic aspects of the simulation.

In our model we use simple integrate-and-fire neurons [5]. For simplicity reasons no leakage has been included in the model. The amount of current leak, in the short period the neuron's state is pooled (simulation of ultra-rapid visual tasks), can be neglected (no rate based coding is present). Neurons receive input spikes and increase their internal activity, until they reach a fixed threshold. The parameters of the neuron fit the biological evidence (Fig. 1).

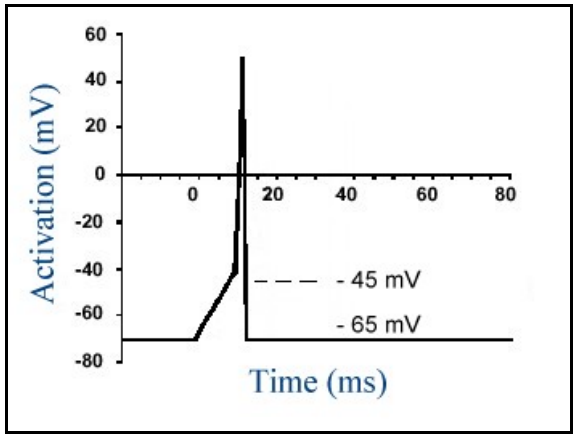


**Fig. 1.** Neuron model. The resting potential is –65 mV and the threshold –45 mV. No refractory period included.

We use rank order coding as neural code and every neuron has 3 parameters: an activation level, a synaptic modulation and an instantaneous sensitivity. The activation level determines the moment of spike and the synaptic modulation and the sensitivity model the fast shunting inhibition.

The update rule states that for every incoming spike, the activation level of the neuron is updated with the synapse weight modulated by the instantaneous sensitivity of the neuron. The sensitivity is then decreased using the synaptic modulation factor [5].

Such a simple neural model requires only 12 bytes for representation when using fixed point arithmetic ($k_{neuron}$ = 12 bytes).

Now we have to deal with synaptic representation. To reduce the amount of information necessary for representing synapses we have to make a commitment at this point. Taking into account that RetinotopicNET is used for simulating the ventral visual pathway we can consider the overwhelming majority of receptive fields as being retinotopic. Retinotopy is that structural property of receptive fields that determines adjacent neurons to receive spikes from adjacent areas in the afferent neural region (Fig. 2). Starting with ganglion cells at lower levels and ending with higher areas, such as the primary visual cortex (V1), most of the receptive fields are retinotopic.

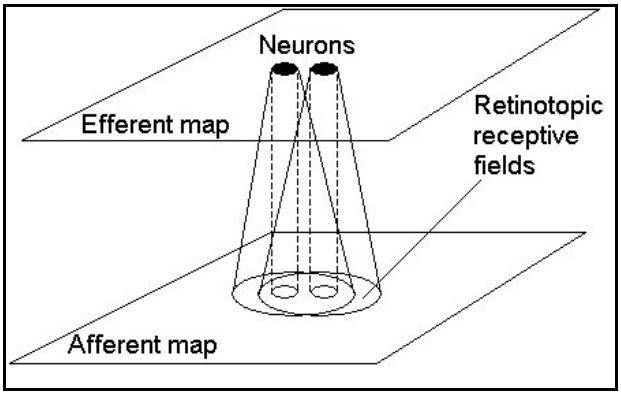


**Fig. 2.** The retinotopy of receptive fields

Considering the homogeneity of receptive fields and the functional resemblance we can group neurons into functional neural maps. All the neurons in one map share the same structure for receptive fields. Thus, there is no need of representing explicitly their retinotopic receptive fields. But we have to say that our remark holds only under the assumption of retinotopy.

We can compress the synaptic information into a synaptic matrix that describes a rule to be used during neural update. The values in the matrix represent synaptic weights. Every pair of neural maps that are interconnected has at least one synaptic matrix associated. The memory consumption overhead can be easily computed by considering a concrete case. Assume there are two interconnected maps that contain $n_1$ and $n_2$ neurons respectively. Let $n_{max}$ be the maximum between $n_1$ and $n_2$ ($n_{max} = \max\{n_1, n_2\}$). Under such circumstances we can write:

$$M_2(n_1 + n_2) = k_{syn} \cdot n_{\max} < k_{syn} \cdot (n_1 + n_2) \quad (2)$$

where:

$M_2(n)$ – is the memory consumption induced by the synapses of n neurons

$k_{syn}$ – represents the average connectivity in terms of number of afferent maps per efferent map (including self-connections and redundancies)

For n neurons, in the worst case, we can maximize:

$$M_2(n) \approx k_{syn} \cdot n \quad (3)$$

Equation (3) gives a worst case estimation for the memory consumption overhead. The $k_{syn}$ is a constant dependent on the particular connections in the model that is simulated using RetinotopicNET. The usage of the synaptic matrix will be described with the dynamics of the model.

To finalize with the memory consumption overhead we can compute the total amount of required memory as in equation (4):

$$M(n) \approx M_1(n) + M_2(n) = (k_{neuron} + k_{syn}) \cdot n$$

Thus:

$$M(n) \approx k_{mem} \cdot n \quad (5)$$

where:

M(n) - is the total memory required for representing n neurons and their synapses

$k_{mem}$ - is a constant that depends on the architecture of the simulated model

The total memory consumption for a given simulated architecture is linearly scalable with respect to the number of neurons in the system.

## 2.2. Solutions to dynamic problems

The dynamic aspects of a simulator are probably more important then the static ones. Committing the system to certain constraints can solve static problems, as we've shown before. However, dynamic aspects influence not only the required hardware but also the applicability of the system in certain engineering fields. For example, most of the systems are desired to work in real applications, where real-time is an issue. There is also a trade-off between the cost of the machine used and the performance of the system. We try to prove that a good design can make monoprocessor systems feasible to real-time simulation of very large networks of integrate-and-fire neurons.

As stated before, there are two aspects that influence the dynamics of the simulator. The first one is related to the computational complexity at the neuron's level. The simpler the neuron's model, the faster is the simulation of its behavior. We try to maintain the neuron's model as simple as possible. RetinotopicNET uses simple, linearly integrating, integrate-and-fire neurons. In addition, rank-order coding has been implemented, in order to account for fast-shunting inhibition [8]. Every time an incoming spike is received on a dendrite the neuron's state is updated using the following rules:

$$U^{(t+1)} \leftarrow U^{(t)} + S^{(t)} \cdot W[current\_synapse] \quad (6)$$

$$S^{(t+1)} \leftarrow S^{(t)} \cdot M \quad (7)$$

where:

$U^{(t)}$ - represents the internal activity of the neuron at time t

$S^{(t)}$ - is the sensitivity of the neuron as resulted from inhibition gathered from the

interneurons. Its value decreases with every spike from 1 towards 0.

M - is a constant synaptic modulation factor. Its value is between 0..1, a value close to 1 inducing a weak shunting inhibition and a value close to 0 inducing strong shunting inhibition

No refractory period is included and a simple comparison of the internal activation with the threshold is done after updating the neuron's state. Such a simple model offers an increased speed of processing.

The second aspect of dynamic simulation involves more complex problems that are related to the information flow in the system and the updates that are necessary for every time step. Many studies showed that most of the neurons, in an asynchronous neural model, are in a steady state and do not require update [4] [2]. The only interesting aspects in the dynamics of integrate-and-fire spiking networks occur due to neural spikes. Therefore, we have no reason to update neurons that do not receive any incoming spikes on their dendrites. It can be stated that the dynamics of an integrate-and-fire neural network are driven by the spike events. Spikes are the most important information to be propagated and processing should be centered only on spike propagation and updates due to spikes.

For all these reasons, an optimal simulator should be event-driven and, in our case, the events are neural spikes.

RetinotopicNET makes use of the event-driven nature of the spiking neural networks. In every time step, a neural map processed its lists of incoming spikes. For every spike in the list we using a simple update technique that determines the neurons that are to be updated as a result of the current spike. An update window is computed using the synaptic matrix and the state of every neuron in the update window is modified. We'll discuss the update window technique later.

After making all the necessary updates, the system generates the new spikes for the current map. The maps connected to the current map on the feed-forward pathway process these spikes. Thus, processing is centered only on necessary updates as a result of incoming spikes. No update is performed for neurons that receive no spikes and this is consistent since we didn't include a refractory period or a leak current in the model.

The number of spikes determines directly the processing effort. For a visual-processing model, extending the system is equivalent to enlarging the size of the maps [5]. As a result, the number of neurons increases. The average number of spikes increases linearly with the number of neurons that generate these spikes (Equation 8).

$$N_{spikes} = k_{spikes} \cdot n \quad (8)$$

where:

$N_{spikes}$ - is the average number of spikes generated during the simulation cycle

$k_{spikes}$ - is a constant that represents the average number of spikes produced by a neuron during its simulation lifetime

n - is the total number of neurons

In a neural model that uses asynchronous spiking $k_{spikes}$ is very reduced yielding a relative stability of the number of spikes generated even when the architecture is extended. We can consider the above dependency as being a sublinear one ( $Nspikes \approx \sqrt{n}$ ). Experimental data should confirm that our assumption is good.

The computational effort does not only depend on the number of spikes but on the average size of receptive fields also. The size of the spike update window is, in most of the cases constant, as derived from the size of the synaptic matrix. However, there are also receptive fields that cover most of the visual field [5] in the ventral visual pathway model. Hence, there is a certain dependency between receptive field size and the number of neurons in the system (that, in the case of visual processing is determined by the size of the stimulus image). The dependency is also sublinear and statistically it can be maximized ( $RF_{size} < \sqrt{n}$ ).

Taking into account all these remarks, we can compute the processing effort, on average, for a typical visual pathway model in simulation. The number of updates on average is given by:

$$N_{upd} \approx N_{spikes} \cdot RF_{size} \approx k_{upd} \cdot n \quad (9)$$

where:

$N_{upd}$ - is the average number of updates as a result of spike generation and propagation during the simulation

$k_{upd}$ - is a constant that captures the architectural properties of the model simulated

n - represents the total number of neurons

In other words the processing effort is, statistically speaking, linearly dependent on the number of neurons in the model. The demonstration is quite empirical but experimental evidence will prove that it is consistent.

### 2.3. Implementation details

Next we'll discuss the way RetinotopicNET makes use of the solutions to static and dynamic problems.

The building blocks of RetinotopicNET are the neural maps. A neural map is a matrix of neurons. Every neuron has three parameters: the activation level, the instantaneous sensitivity and the synaptic modulation. Additionally, a map has two associated lists: a spike list and a list of the afferent maps (Fig. 3).

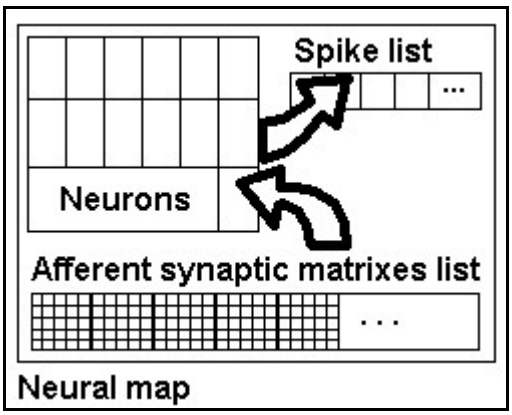


**Fig. 3.** The structure of a neural map

For every afferent map, a synaptic matrix is memorized in the synaptic list. During simulation, the spikes from the afferent maps are inspected. For each spike, the system selects the appropriate synaptic matrix (corresponding to the afferent map that generated the spike). The synaptic matrix is used for computing the spike's update window, in other words the neurons in the current map that need to be updated as a result of the incoming spike.

There are two cases to handle. The first case consists of identical sized maps. Using the synaptic matrix and the offset from the spiking position to the position of the neuron to be updated, the exact synaptic weight is selected from the matrix and used in the update (Fig. 4).

The second case is more complicated and appears when the afferent and efferent maps have different sizes. In this case the system handles both overlapping and non-overlapping receptive fields. The mechanisms allow RetinotopicNET to achieve neural zooming [6].

The information flow inside a neural map is shown in Fig. 1. Spikes are processed from the afferent maps using the correspondent afferent synaptic matrix. The spikes resulted from neural update are inserted into the spike list generated during the current iteration. Simulation is iterative and updates occur at fixed time slices. For every iteration spikes are being fed into the system and then every map is updated in an ascending manner.

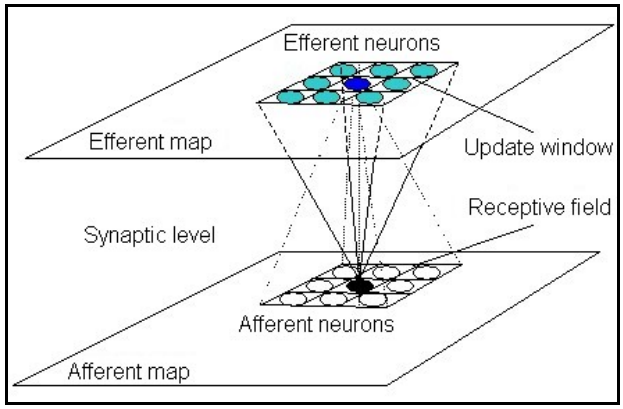


**Fig. 4.** Determination of the update window

We mentioned that the simulator uses rank-order coding as implementation of the fast shunting inhibition [8]. Under such circumstances the system has to deal with spikes that arrive theoretically simultaneously. Instead, every incoming spike desensitizes the target neuron. The order of spike arrival is therefore essential. The problem is known as the "ex aequo" problem [9]. The ideal solution is to keep the neuron's sensitivity constant for incoming spikes that are simultaneous and decrease it only once at the end of the spike sequence. But this approach would be very time consuming. Instead, RetinotopicNET minimizes the error resulted by equally pooling every afferent map during simulation. In this way, no map is penalized and all have equal chances. In other words, instead of processing the whole spike list for each afferent map, the system peeks one spike at a time for every afferent map and completes a pooling cycle over all afferent maps. Then the cycle is repeated until all the afferent spikes had been inspected.

Another problem to be solved is that of self-connection or lateral interaction between neurons in the same map. To deal with this, RetinotopicNET first processes the incoming spikes from afferent maps and then completes an additional cycle for lateral interaction. As a result of inhibition some spikes are pooled off the spike list. In order to achieve this, the system saves and then flushes the spike list and uses it for lateral interaction. The spikes saved are filtered and then merged, with the new spikes generated, into the final spike list.

## 3. Results and discussion

The first step in testing RetinotopicNET was to prove that processing time is also linearly scalable with respect to the number of neurons. We tried to prove that the assumptions in section 2.2. have experimental support. The testing results in Fig. 5. prove that the processing time varies approximately linearly with the number of neurons in the simulation.

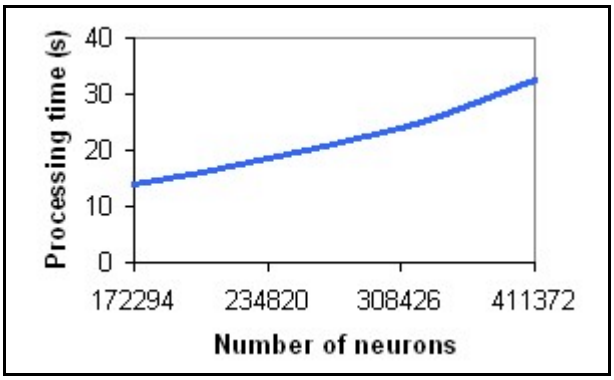


**Fig. 5.** Evolution of processing time for different numbers of neurons in the model on a Pentium® 200 MHz processor system.

We tested RetinotopicNET for simulation of the ventral visual pathway, responsible for object recognition in the brain. The architecture involves several millions of neurons and billions of synapses. Using the model of the ventral visual pathway we achieved complex object recognition in cluttered environment [5].

Simulation times vary with the complexity and size of the input image. The model can be easily calibrated by setting the number of spikes that are propagated from the retina level. Ultra-rapid visual categorization occurs before all the retina-generated spikes are propagated to the infero-temporal cortex. When only 20% of the neurons spiked at the retina level, the recognition occurs and the simulation can be stopped. The calibration of the system consists of setting the synaptic gain for the ganglion cells at the retina level [5]. Based on this calibration a trade-off between number of neural spikes and accuracy of recognition can be obtained. The tuning of the system was done empirically and taking into account that training had also been supervised. On a Windows 2000® platform with a single Pentium® 200Mhz MMX processor RetinotopicNET simulates 2 millions of neurons and 3 billions of synapses for the given model in a time magnitude of 3 to 5 minutes. With further calibration the system had been able to perform real-time processing for object recognition [5].

An improvement is possible by associating an absolute time of spiking to every spike generated. The absolute time can be computed using the activation of the neuron and the incoming excitation / inhibition. Such a strategy would offer the system a continuous time domain for simulation and could eliminate discretization errors.

*References:*


[1] **Bower, J. M., & Beeman, D.** *The book of GENESIS.* New York: Springer-Verlag, 1998.

[2] **Delorme, A., et al.** SpikeNET: A simulator for modeling large networks of integrate and fire neurons. In J. M. Bower (Ed.), *Computational neuroscience: Trends in research 1999, neuro-computing* (Vols. 26–27, pp. 989–996). Amsterdam: Elsevier Science.

[3] **Hubel, D. & Wiesel, T.,** Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *J. Neurophysiol.,* No. 28, 1965, pp. 229–289.

[4] **Mattia, M. & Del Giudice, P.** Efficient Event-Driven Simulation of Large Networks of Spiking Neurons and Dynamical Synapses, *Neural Computation,* 12, 2000, pp. 2305–2329.

[5] **Mureşan, R.**, Complex Object Recognition Using a Biologically Plausible Neural Model, *Proceedings of the 2nd WSEAS International Conference on Robotics, Dis-tance Learning and Intelligent Communication Systems,* Skiathos, Greece, Sept. 25-28, 2002.

[6] **Mureşan, R.**, Visual Scale Independence in a Network of Spiking Neurons, *Proceedings of the 9th International Conference on Neural Information Processing*, Singapore, Nov. 18-22, 2002.

[7] **Thorpe, S., Fize, D. & Marlot, C.,** Speed of processing in the human visual system, *Nature, 381*(6582), 1996, pp. 520-522.

[8] **Thorpe, S.J. & Gautrais, J.,** Rank order coding. In J. Bower, *Computational neuro-science: Trends in research,* (pp. 113-118), New-York: Plenum Press, 1998.

[9] **Veneau, E.,** Codage Impulsionnel Par Rang Et Apprentissage, *Cerco*, 1996.