ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Saliency extraction with a distributed spiking neural network

Sylvain Chevallier[1] and Philippe Tarroux[1,2] and Hélène Paugam-Moisy[3]

1- LIMSI - UPR CNRS 3251
Orsay - France

2- École Normale Supérieure
Paris - France

3- Institute for Cognitive Science - UMR CNRS 5015
Lyon - France

Abstract. We present a distributed spiking neuron network (SNN) for handling low-level visual perception in order to extract salient locations in robot camera images. We describe a new method which reduce the computional load of the whole system, stemming from our choices of architecture. We also describe a modeling of post-synaptic potential, which allows to quickly compute the contribution of a sum of incoming spikes to a neuron's membrane potential. The interests of this saliency extraction method, which differs from classical image processing, are also exposed.

1 Introduction and framework

The bio-inspired paradigm aims at adapting for computer systems what we understand from the evolution-selected solutions. Such transfers have already been made successfully, *e.g.* the Brook's robots [1] and his subsumption architecture, Floreano's work [2] or the research reviewed by Steels [3].

We propose a distributed SNN[1] for handling low-level visual perception (section 2). It is known that the visual system has limited capacities and that organisms have developed attentional mechanisms, which select stimuli in complex visual scenes. Psychologists make a useful distinction between attention and pre-attention processing : the former is a top-down process, *e.g.* involving task-dependent or context-dependent influences. The latter is a bottom-up (BU) process, *i.e.* data-driven, on which we will focus in this work. Many models of selective visual attention with BU process have been proposed [4, 5, 6], a detailed review can be found in [7]. Yet, previous work rely on classical image processing for extracting saliencies whereas we try to exploit the possibilities provided by SNN models (section 2.1). SNN are a recent kind of neural model, detailed in [8, 9], which take into account temporal dynamics. Although the expected capacities of SNN model are still debated [10, 11], they provide a temporal coding which begins to be broadly used (*i.e.* [12, 13]). The SNN we consider is a distributed system, described in Section 2.2 and the results are shown in Section 3. They are based on images acquired by the camera of a robot in real environment.

[1]SNN: Spiking Neuron Network

2 Distributed Spiking Neuron Network

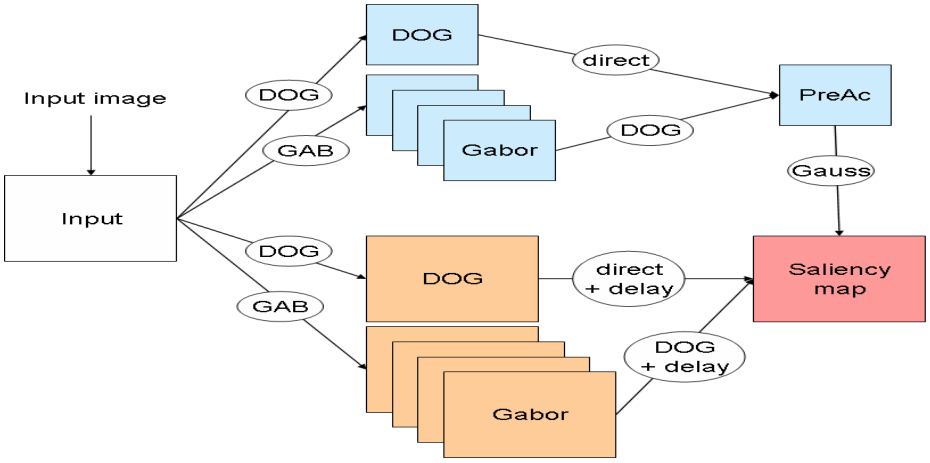

Fig. 1: Organization of architecture, a set of feedforward connected SNN 2D maps (colored squares), which extract saliencies from an input image. Connection masks are illustrated by oval: Gauss, DOG and GAB are explained in Section 2.3. Direct means connections between pairs of neurons with the same coordinates and delay means fixed delayed synapses.

The architecture is designed to extract the saliencies of a complex visual scene and can be considered as a step towards a complete robot neural controller system. It is composed of a set of feedforward connected maps (2D layer spiking neuron networks) and can handle at the same time different neural models. The architecture is divided into two main pathways for handling processing on high (resp. low) spatial frequencies, bottom (resp. top) boxes on Fig. 1. We use low frequencies analysis for selecting conspicuous areas, which are gathered on a pre-activation map (Fig. 1: PreAc). A salient location is detected (Fig. 1: saliency map) when there is a conspicuous point, extracted from high spatial frequency maps, in a conspicuous area. This is a simplified adaptation of a multi-scale processing (see [7] for examples).

This two way separation is inspired from the parvo and magno-cellular channels in the primate central nervous system. One way quickly processes a rough signal in low frequencies and a much slower way processes fine-grained details. The fact that spiking neurons are good coincidence detectors makes them efficient to detect conjunctions of high and low spatial frequency information. Thus the proposed architecture extract saliencies only if the conjunction of high and low frequency information is met. A pixel is considered to be salient if a neuron of the saliency map emits a spike and it occurs only if the neuron receive concomitant spikes from both channels.

2.1 Neural maps

The base units (the "basic bricks") of this architecture are neural maps. They are structured in 2D layer of NxM neurons ($\frac{N}{2}$x$\frac{M}{2}$ for the low spatial frequency neural maps), where NxM is the input image size in pixels. The neural map topology can be viewed as a sort of "retinotopic" organization, because the topological properties of the input image are preserved on neural maps.

The first map (Input on Fig. 1) creates an input current proportional to each pixel luminance of the input image for each neuron. This map converts pixel luminance into a discharge rate code: highly luminous pixels are translated into high input currents, so spikes are triggered quickly, forming fixed frequency spike trains. This activity is spread towards other maps and initiates a stable dynamic in the whole network.

The spiking neuron model used in neural maps is a *linear leaky integrate-and-fire* (see eq. 1), which is simpler than Gerstner's *SRM* [9] but still close to biological neurons. The membrane potential is gouverned by the following differential equation:

$$\begin{cases} \tau \dot{V} = g_L(V - E_L) + PSP(t) \text{ , if } V \geq \vartheta \\ \text{spike otherwise} \end{cases} \tag{1}$$

Where $PSP(t)$ represents the influence of incoming spikes on the membrane potential (called post-synaptic potentials or PSPs) and τ, g_L, ϑ and E_L are parameters of the leaky integrate-and-fire neural model. The evolution of a PSP is commonly described by an α-function, like $\frac{t}{\tau} e^{\frac{-t}{\tau}}$. However, there is no fast method for computing the influence of a sum of α-functions. So we use a gaussian difference model for describing the time course of a PSP, see eq. 2.

$$\begin{cases} \dot{q}_0(t) = -k_0 q_0(t) \\ \dot{q}_1(t) = k_0 q_0(t) - k_1 q_1(t) \end{cases} \tag{2}$$

A gaussian difference PSP model can have a time course very similar to an α-function PSP model and it has been shown that α-function model is a particular case of the gaussian difference model [9]. Assuming that contributions of PSPs are additive, we can easily compute the contribution of a sum of PSPs on the membrane potential (the whole demonstration is not detailled here for space constraints). With an α-function model, one must compute the influence of each PSP to the membrane potential whereas with the gaussian difference model, the membrane potential can be computed in one step.

2.2 Distributed system

Spiking neuron models are both time-consuming, because they are based on differential equations, and handle discrete information, the spikes. So a distributed simulation is well suited and can greatly improve performance for large networks [14], *e.g.* for the real-time control of a mobile robot. The neural maps are the

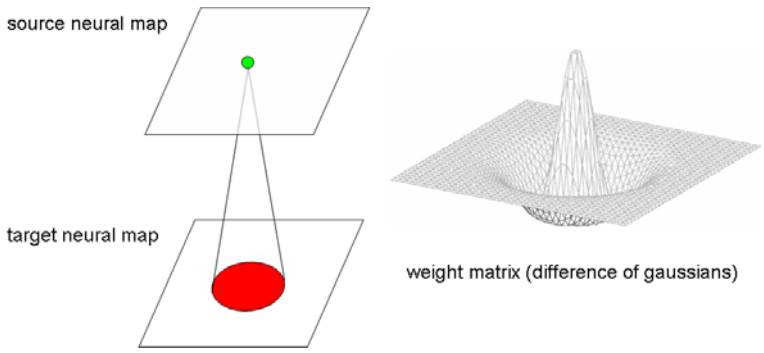

Fig. 2: A neuron (in green or light grey) of the source neural map emits a spike and sends PSPs to all red neurons (or dark grey) on the target map. The weight of each PSP is determined by the weight matrix, represented on the right. The weight matrix takes its values from image processing filter, here a difference of gaussians, which is a constrast detector.

smallest units of the distributed system presented here. Each one run independently and must be assigned to a separate computing node. The distributed computation lowers the time needed to extract saliency compared to the non distributed version of the program. Due to the fact that the distributed system integrate more than 100,000 neurons, it is still time consuming since it is slower than classical image processing methods. However, it provides some temporal information (explained in 3), which is difficult or even impossible to obtain with classical image processing.

Distributed SNN faces a temporal synchronization problem since it can be seen as a set of differential equations spread on several computing nodes: the logical time must be homogeneous across all nodes. At time t, how can we handle information occuring at time $t-1$? It is possible to avoid a unique time controller, as demonstrated in [14], but we choose a simpler system: for each time step, each neural map sends a message to all neural maps with which it has outgoing connections and waits for all neural maps with which it has incoming connections before updating. Thus all nodes keep an homogeneous logical time. The cost of this method is null if there are always outcoming messages on all neural maps, *e.g.* high activity in the network.

SNN can benefit from an event-driven simulation, because neurons can be seen as some sort of discrete-making devices, which convert continuous inputs into a sequence of discrete impulses. If a neuron is not integrating a PSP, it will not be active. Thence a simulation doesn't need to spend time to check up inactive neurons, and a neuron is inactive until it is reached by an incoming PSP. Thus the presented architecture is led by an event-driven implementation and only active neurons are processed.

2.3 Connection masks

When a neuron on the source neural map emits a spike, it sends PSPs to NxN neurons on the target neural maps, as shown on Fig. 2. As all neurons of a single map have the same connection pattern, a generic projective connection, called *mask* is defined. Masks are NxN weight matrices with delays, weights and delays are static, so they can be defined at the beginning of the simulation. The weight matrices are similar to matrix filters used in image processing. The filters used here are contrast detectors (differences of gaussians or DOG) and orientation detectors (gabor wavelets or GAB). In an image processing approach, the resulting image is obtained by the convolution of a matrix filter and an image, and each pixel is processed in an order determined by the algorithm. With connection mask, the pixels are processed in an order determined by the input, from the more luminous to the darkest ones. This speed up the processing as only pixels which are luminous enough are processed: the darkest pixels do not trigger spikes.

Furthermore, connection mask provides an easy way to reduce the number of messages to transmit between distributed nodes. When a neuron fires, only one spike is sent to the target map through the communication channel, the PSPs being generated locally on the target map. This method reduces the size and the numbers of messages sent in the distributed system.

3 Results and conclusion

Fig. 3: Input image (left), saliencies extracted by classical processing (center), saliencies extracted by our architecture (right). The numbers on the latter figure specify the temporal order of occurence.

The results obtained in this study are based on images acquired during a robot ride in our laboratory. The images were reduced from 320x240 pixels to 76x56 pixels and only on the luminance information was considered. Figure 3 shows a prototypical example of our results, the algorithm used for comparison is a combination of gabor and DOG multi-scale processing inspired from Itti's model [7]. The salient locations appear in black on the center image (classical image processing) and on the right image (the presented architecture). Although the salient locations are not exactly the same, the main saliencies are qualitatively identical.

The presented architecture made time-dependent classification of the salient location. The resulting locations presented on Fig. 3 are ordered by decreasing saliencies (small figures). A black dot symbolize a neuron emitting a spike. The earlier a spike is triggered, the more salient is the location, so saliencies are classified using a rank order coding as described in [15]. It is a useful improvement since in previous image processing work the classification of salient locations is difficult to obtain: *e.g.* in Itti's model, the most salient location is chosen by a winner-takes-all algorithm then an inhibition-of-return mechanism prevent the same location to be chosen again during a certain time. The presented architecture classify salient location at no cost because of the temporal processing made by the SNN.

We have presented here an architecture which allows the extraction of salient locations, using distributed SNN. The salient locations detected are close to those detected by classical image processing. The presented system takes advantage of distributed computing but is still slower than image processing methods. We propose an improvement for computing the influence of a sum of PSPs and the described architecture deal with synchronization problem of distributed SNNs, due to the task constraints.

References

[1] R. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[2] D. Floreano, J. Godjevac, A. Martinoli, F.Mondada, and J-D. Nicoud. *Advances in Intelligent Autonomous Agents*, chapter Design, Control, and Applications of Autonomous Mobile Robots. Kluwer, 1998.

[3] L. Steels. The artificial life roots of artificial intelligence. *Artificial Life*, 1(1):1–86, 1994.

[4] A. Treisman and G. Gelade. A feature-integration theory of attention. *Cog. Psy.*, 12(1), 1980.

[5] C. Koch and S. Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human Neurobiology*, 4(4):219–227, 1985.

[6] J. Tsotsos, S. Culhane, W. Yan Kei Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial intelligence*, 78:507–545, 1995.

[7] L. Itti, G. Rees, and J. Tsotsos, editors. *Neurobiology of Attention*, chapter Models of Bottom-Up Attention and Saliency. Elsevier, 2005.

[8] W. Maass and C. Bishop, editors. *Pulsed Neural Networks*. MIT-Press, 1999.

[9] W. Gerstner and W. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

[10] R. Legenstein, C. Näeger, and W. Maas. What can a neuron learn with spike-time-dependent plasticity? *Neural Computation*, 17(11):2337–2382, 2005.

[11] J. Sima and J. Sgall. On the nonlearnability of a single spiking neuron. *Neural Computation*, 17(12):2635–2647, 2005.

[12] L. Perrinet. Finding independent components using spikes : a natural result of hebbian learning in a sparse spike coding scheme. *Neural Computation*, 3(2):159–175, 2003.

[13] C. Brody and J.J. Hopfield. Simple networks for spike-timing-based computation, with application to olfactory processing. *Neuron*, 37:843–852, 2003.

[14] A. Mouraud, H. Paugam-Moisy, and D. Puzena. A distributed and multithreaded neural event driven simulation framework. In *Proc. of PDCN'06, Parallel and Distributed Computing and Networks*, pages 212–217. ACTA-Press, 2006.

[15] R. VanRullen, J. Gautrais, A. Delorme, and S. Thorpe. Face processing using one spike per neuron. *Biosystems*, 48:229–239, 1998.