A Model of Neural Inspiration for Local Accumulative Computation

José Mira[1], Miguel A. Fernández[2], Maria T. López[2], Ana E. Delgado[1], and Antonio Fernández-Caballero[2]

[1] Departamento de Inteligencia Artificial
Facultad de Ciencias y E.T.S.I. Informática, UNED,
28040 - Madrid, Spain
{jmira,adelgado}@dia.uned.es
[2] Departamento de Informática
E.P.S.A., Universidad de Castilla-La Mancha,
02071 – Albacete, Spain
{miki,mlopez,caballer}@info-ab.uclm.es

Abstract. This paper explores the computational capacity of a novel local computational model that expands the conventional analogical and logical dynamic neural models, based on the charge and discharge of a capacity or in the use of a D flip-flop. The local memory capacity is augmented to behave as an *S* states automaton and some control elements are added to the memory. The analogical or digital calculus equivalent part of the balance between excitation and inhibition is also generalised to include the measure of specific spatio-temporal features over temporal expansions of the input space (dendritic field). This model is denominated as accumulative computation and is inspired in biological short-term memory mechanisms. The work describes the model's general specifications, including its architecture, the different working modes and the learning parameters. Then, some possible software and hardware implementations (using FPGAs) are proposed, and, finally, its potential usefulness in real time motion detection tasks is illustrated.

1 Introduction

The most usual analogical models in neural computation are of a static nature. Once the input values in an instant, $\bar{x}(t)$, and the values of the weights, $\bar{\omega}(t)$, are known, the output value in that instant, $\bar{y}(t) = \bar{\omega}(t) \cdot \bar{x}(t)$, is obtained. Nevertheless, one important part of the biological processes and of the proper computation are rather of a dynamic nature; that is to say, they are models dependent of time where the response, $\bar{y}(t)$, is a function of the inputs and responses in earlier instants, $\left\{\bar{x}(t - K_1 \cdot \Delta t), \bar{y}(t - K_2 \cdot \Delta t)\right\}$. In order to model these dynamic nets a set of state variables described by a first order differential equation $\tau_j \frac{dy_i(t)}{dt} = -y_i(t) + h_j$, are

R. Moreno-Díaz and F. Pichler (Eds.): EUROCAST 2003, LNCS 2809, pp. 427–435, 2003.

introduced, such that in the stationary case variable $\bar{y}(t)$ reaches its equilibrium value (h_j) with a time constant τ_j.

When adding the effect of the inputs $\{x_i(t)\}$, the linear part of the expression of a dynamical neural model known as leaky integrator is gotten. This means that the value and the sign of the state variable depend on the excitation and inhibition in the receptive field of the calculus element:

$$\tau_j \frac{dy_j(t)}{dt} = -y_j(t) + \sum_{i \in V_j} w_{ji} \cdot x_i(t) + h_j$$

In this case the influence of the temporal component of the calculus (the analogical memory) is physically represented by means of charge and discharge processes of a capacitor [10].

On the other hand, in digital models of neural networks, local memory is introduced by means of a D flip-flop that represents the effect of the synaptic delay [11]. In this case the computational model is a modular sequential circuit (a modular automaton) in which each calculus element ("neurone") is a universal two states automaton, which may calculate any logical function of its inputs and of the proper and other neurones outputs in the previous instant,

$y_j(t+\Delta t) = \sum_{i=0}^{2^{N+M}-1} \omega_{ij}(t) \cdot m_i(t)$, where $\omega_{ij}(t) \in \{0,1\}$, are the binary weights and $m_i(t)$ are the minterms, $m_i = x_1^{\alpha} \cdot x_2^{\beta} \cdots x_M^{\mu} \cdots y_N^{\gamma}$ *for* $i = \alpha\beta \cdots \mu \cdots \gamma$,

using Gilstrap notation: ($x^0 = \bar{x}$, $x^1 = x$)

This paper explores the computational capacity of a novel local computational model that expands the conventional analogical and logical dynamic neural models, based on the charge and discharge of a capacity or in the use of a D flip-flop. The local memory capacity is augmented to behave as an *S* states automaton and some control elements are added to that local memory. The analogical ($W^T \bar{x}(t)$) or digital ($\sum \omega_{ij}(t) \cdot m_i(t)$) calculus equivalent part of the balance between excitation and inhibition is generalised to include any pre-processing not related to learning where spatio-temporal features of the stimuli are calculated over temporal expansions of the input space. This expansion with a FIFO memory structure represents the computational features of the receptive field, which make computationally homogeneous the data fields coming from different time intervals. The part corresponding to the delay management is also generalised by substituting it by an *S* states automaton with a reversible counter structure (or a RAM memory), where the increment and decrement of its content is programmable. This model is denominated as accumulative computation.

The rest of the paper is organised in the following way. Section 2 describes the model's general specifications, including its architecture, the different operating modes and the learning parameters. Afterwards, in sections 3 and 4 some software and hardware (using FPGAs) implementations are proposed. Section 5 illustrates the potential usefulness of this local computational model in real time motion detection tasks.

2 The Model's Functional Architecture

Figure 1 shows the accumulative computation model's block diagram. The model works in two time scales, a macroscopic one, t, associated to the external data sequence to be processed by the net, and a microscopic one, τ, internal, associated to the set of internal processes that take place while the external data (an image, for instance) remain constant. The model contains the following elements:

1. A temporal extension of the input space (a FIFO memory) that permits to access the value of the inputs in various successive time instants.
2. A module of spatio-temporal features extraction over that input expansion. The measured feature is binarised and, from this moment on, the temporal accumulation of its persistency on that data field is calculated.
3. A module that calculates the increment or decrement value, ($\pm\delta Q$), of the activity state of that property as a function of its value in that instant, $Q(t)$, of the accumulated value in the previous instants and of the accumulation mode selected in the control unit.
4. An accumulation module of reversible counter type or RAM memory, which stores the new persistency state of the selected feature.
5. A control module of the accumulation mode, which receives inputs from the programming and learning modules, and controls the operation of state changing of the memory from the calculus of increments or decrements, $\pm\delta Q$, on the previous value. There are three general operating modes for the model: (I) Initialisation, (II) calculation and (III) reconfiguration (learning). During the calculation mode, and in accordance with the temporal sequence of values $p(x,y;t)$ measured on the input data $I(x,y;t)$, one of the following processes is activated: (1) Gradual charge, (2) abrupt charge, (3) gradual discharge, (4) abrupt discharge or (5) stand-by. The parameter values that specify the charge and discharge processes (Q_{max}, Q_{min}, $+\delta Q$, $-\delta Q$) are introduced into the model during the initialisation phase and are modified during the learning phase.
6. A module of supervised learning, which enables to adjust the value of charge and discharge parameters to the shape, size and velocity features of the objects of interest that appear in the image sequences.
7. A programming module used to configure the control mode and to specify the temporal expansion of the input space and the shape of the receptive field. This way it is possible to specify the spatio-temporal property that we want to highlight alter accumulating its persistency.
8. A temporisation module consisting of a master clock, which generates the pulse train that controls the local time ("microscopic") used to calculate the value and sign of the accumulation state change and the transition to the new charge state, as well as the production of the response of the unit that passes to a FIFO for its distribution to the neighbouring modules. While the internal calculus is performed, data of the input space remain constant, controlled by the "macroscopic" clock resulting from having divided by n the frequency of the master clock.

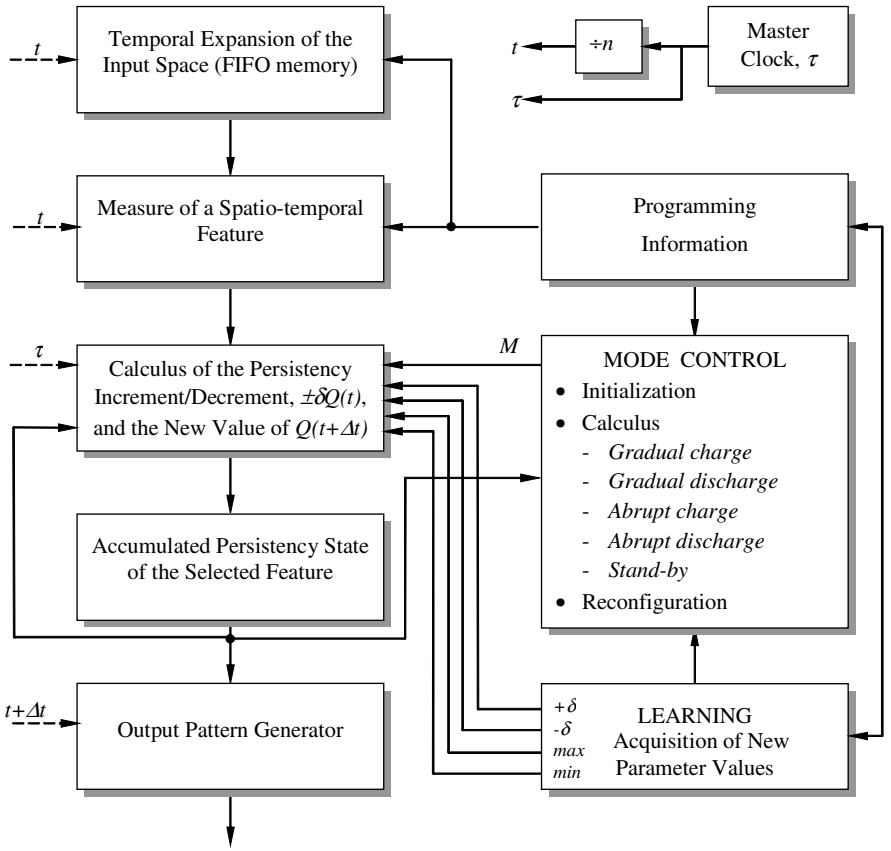


Fig. 1. Accumulative computation architecture

3 Software Simulation

In figure 2 the accumulative computation model's behaviour is shown in one-dimensional and very easy situations. Let us suppose that values of $I(x,y;t)$ correspond to an indefinite sequence of images where several objects are moving. Let us also suppose that the measured property, $p(x,y;t)$ is simply the result of the binary threshold of image $I(x,y;t)$. Then, the control mode compares values of $p(x,y;t)$ in two successive instants, interpreting that $p(x,y;t)=1$ means that there is a moving object over pixel (x,y) at t and that $p(x,y;t)=0$ means there is no mobile. Thus, changes $p(t-\Delta t)=0 \Rightarrow p(t)=1$ mean that a moving object has entered that unit's receptive field. If $p(t-\Delta t)=1$ and $p(t)=0$, a mobile has quitted the receptive field (RF); if both are zero, there is no mobile over the RF, and, finally, if both are one, there is a moving object crossing over the RF. For this property, the evolution of charge and discharge of its persistency is shown in figure 2 for some modalities of use selected.

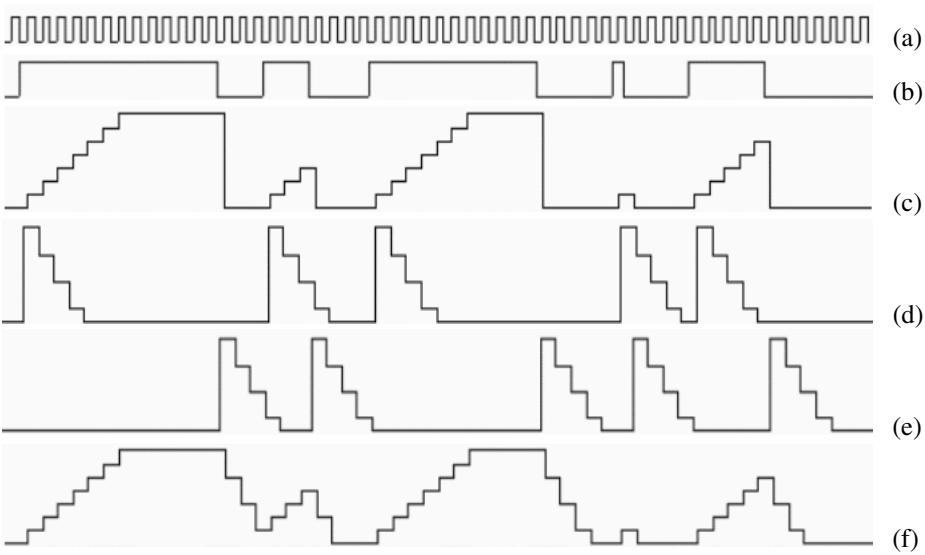


Fig. 2. Illustration of the accumulative computation model used for the easy case of binary threshold of an image. (a) Macroscopic clock *t*. (b) *p(t)*. (c) *Q(t)* in LSR modality. (d) *Q(t)* in input modality. (e) *Q(t)* in output modality. (f) *Q(t)* in charge/discharge modality

Figure 2c shows the behaviour of the accumulative computation model in a modality called LSR (length speed relation) [1]. This modality has been studied and used for the classification of moving objects from this relation [2]-[4].

```
if p(t) == 1
   then
     begin
       Q(t) = Q(t-Δt) + δQ;
       if Q(t) > Qmax then Q(t) = Qmax;
     end
   else Q(t) = Qmin;
```

Figures 2d and 2e show the operation of the proposed model in input and output modalities, respectively. Both options enable to perform a later calculus of characteristic motion parameters, such as velocity and acceleration [5]. The first one of these modalities offers information at the tail of the moving objects, whereas the second one does it at the front of motion. For the output modality we have:

```
if ((p(t-Δt) == 0) && (p(t) == 1))
   then Q(t) = Qmax
   else
     begin
       Q(t) = Q(t-Δt) - δQ;
       if Q(t) < Qmin then Q(t) = Qmin;
     end;
```

In input modality we have:

```
if ((p(t-Δt) == 1) && (p(t) == 0))
   then Q(t) = Qmax
   else
     begin
       Q(t) = Q(t-Δt) - δQ;
       if Q(t) < Qmin then Q(t) = Qmin;
     end;
```

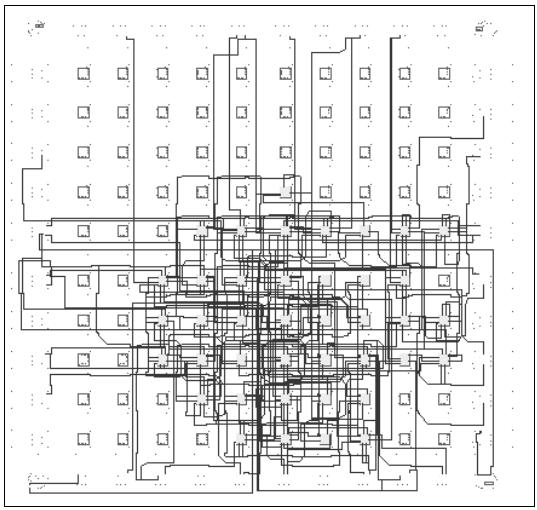
Finally, the more general charge/discharge modality is shown (figure 2f). This one has already been successfully used in some previous papers of the authors of this work [6]-[9]. These papers are about moving objects detection, classification and tracking in indefinite image sequences.

```
if p(t) == 1
   then
     begin
       Q(t) = Q(t-Δt) + δQ;
       if Q(t)> Qmax then Q(t) = Qmax;
     end
   else
     begin
       Q(t) = Q(t-Δt) - δQ;
       if Q(t)< Qmin then Q(t) = Qmin;
     end;
```

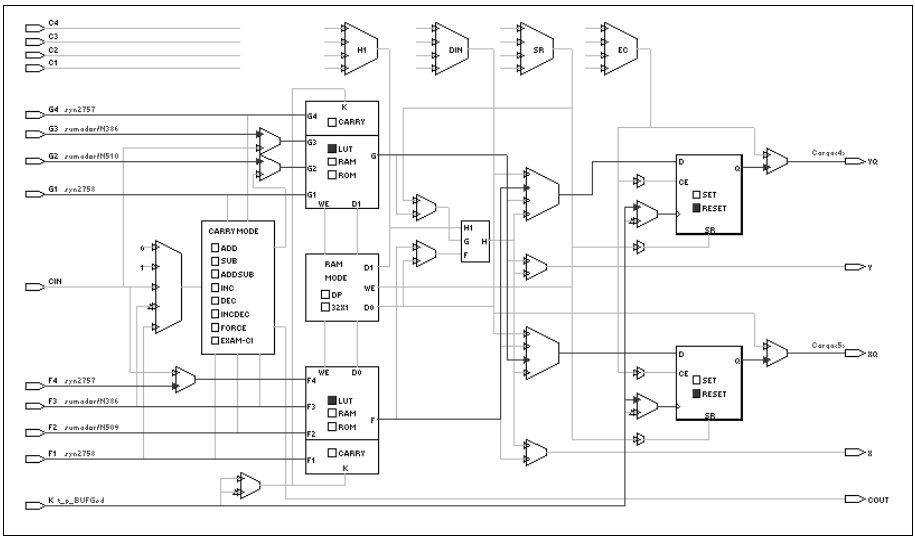
4 Hardware Implementation

The very nature of the intended calculus and the need for reconfiguration demanded by the accumulative computation model advises its implementation by means of a programmable sequential logic (e.g. field programmable gate arrays, FPGAs). These circuits contain a high number of reconfigurable identical logical modules (configurable logical blocks, CLBs) at the modules internal structure level as well as at interconnection level, and, in both cases, by simple modification of the content of a set of RAM memory cells. As an example, figure 3 shows the result of the accumulative computation model's hardware implementation on a Xilinx 4000E chip, concretely the X4003E.

Figure 3a shows the result of programming in VHDL language, and synthesising and implementing for an FPGA X4003E. In this implementation the number of different charge values *Q(t)* has been restricted to 256. Notice that only the four modalities previously described have been implemented. Figure 3b shows in detail one of the CLBs that form the chip. The design statistics offered by the Xilinx implementation tool show a total equivalent gate count for design of 588, whereas the performance of the chip shows a mimimum period value of 32.712 ns.



(a)



(b)

Fig. 3. Hardware implementation with FPGAs. (a) FPGA X4003E chip. (b) Detail of a CLB.

5 Real Time Motion Detection Tasks

By configuring adequately each model's modules (input space, measured property, output generator, control circuit and clocks), we do it suitable for a family of applications, remaining invariant its conceptual structure: to measure a property and store it in a local memory with the possibility of forgetting.

In this paper its usefulness in moving objects detection, classification and tracking tasks in an indefinite image sequence is illustrated. A distinctive characteristic of the approach given by this model is the possibility to compute in real time, as a consequence of the ergonomic character of the process. Figure 4 shows some of the model's capabilities introduced so far.

Indeed, figure 4 shows the model output for three significant examples. All three examples are the result of applying our model to a same image sequence, namely TwoWalkNew, downloaded from University of Maryland Institute for Advanced Computer Studies, copyright © 1998 University of Maryland, College Park. The pure output is shown on the first column for each example. The second column shows the result superimposed on the input image. The results of silhouette detection are drawn in figure 4a, motion detection in figure 4b, and direction detection in figure 4c. In this last example notice that motion direction is shown by means of the intensity of the colour. Direction has to be interpreted going from clearer to darker grey colour.

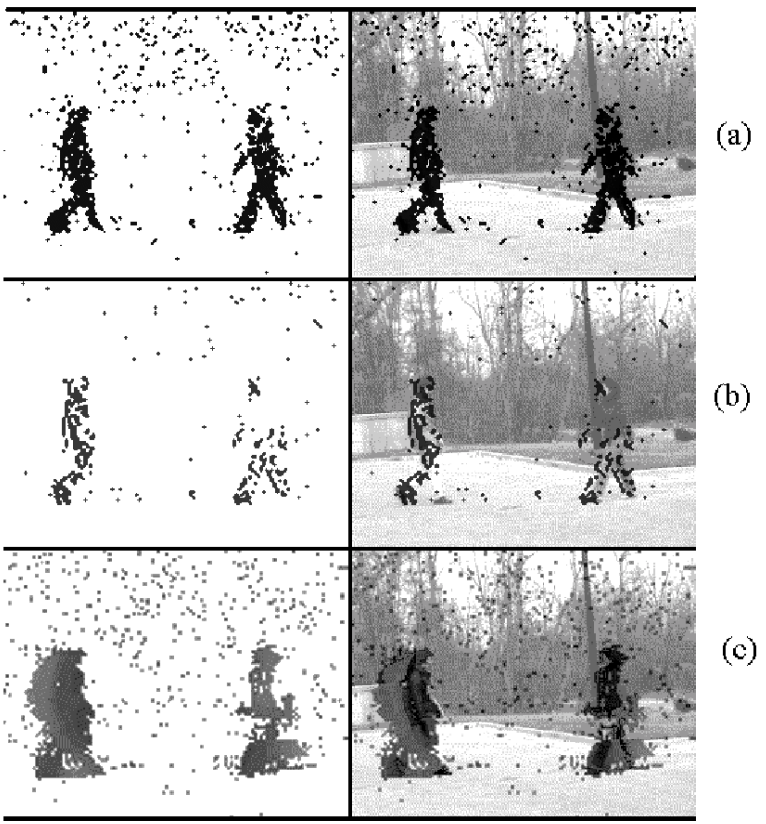


Fig. 4. Some capabilities of the accumulative computation model. (a) Silhouette detection. (b) Motion detection. (c) Direction detection

6 Conclusions

A calculus model that is modular, dynamic, of fine grain, partially self-programmable by supervised learning, and able to be integrated in a parallel architecture, has been introduced in this paper. It might be called "neuronal", but it seems to us more adequate to consider it as a model of local calculation inspired in biological memory mechanisms. We have increased the capacity for local calculus, the memory, the features extraction as a pre-processing and the generation of output patterns. Thus, the model converts into a real time processing architecture of spatio-temporal information, based on the controlled management of a local memory.

Lastly, this paper has shown the usefulness of the accumulative computation model in artificial vision. Its use in tasks such as velocity and acceleration obtaining, moving objects detection, silhouettes detection and selective visual attention generates efficient and robust systems with competitive performances compared to any model used nowadays.

References

1. Fernández, M.A., Fernández-Caballero, A., López, M.T., Mira, J.: Length-Speed Ratio (LSR) as a characteristic for moving elements real-time classification. Real-Time Imaging 9:1 (2003) 49–59
2. Fernández, M.A., Mira, J.: Permanence memory: A system for real time motion analysis in image sequences. IAPR Workshop on Machine Vision Applications, MVA'92 (1992) 249–252
3. Fernández, M.A., Mira, J., López, M.T., Alvarez, J.R., Manjarrés, A., Barro, S.: Local accumulation of persistent activity at synaptic level: Application to motion analysis. In: Mira, J., Sandoval, F. (eds.): From Natural to Artificial Neural Computation, IWANN'95, LNCS 930. Springer-Verlag (1995) 137–143
4. Fernández, M.A., Fernández-Caballero, A., Moreno, J., Sebastián, G.: Object classification on a conveying belt. Proceedings of the Third International ICSC Symposium on Soft Computing, SOCO'99 (1999)
5. Fernández, M.A.: Una arquitectura neuronal para la detección de blancos móviles. Unpublished Ph.D. dissertation (1995)
6. Fernández-Caballero, A., Mira, J., Fernández, M.A., López, M.T.: Segmentation from motion of non-rigid objects by neuronal lateral interaction. Pattern Recognition Letters 22:14 (2001) 1517–1524
7. Fernández-Caballero, A., Mira, J., Delgado, A.E., Fernández, M.A.: Lateral interaction in accumulative computation: A model for motion detection. Neurocomputing 50C (2003) 341–364
8. Fernández-Caballero, A., Fernández, M.A., Mira, J., Delgado, A.E.: Spatio-temporal shape building from image sequences using lateral interaction in accumulative computation. Pattern Recognition, in press (2003)
9. Fernández-Caballero, A., Mira, J., Férnandez, M.A., Delgado, A.E.: On motion detection through a multi-layer neural network architecture. Neural Networks, accepted (2003)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall (1999)
11. Moreno-Díaz, R.: Realizability of a neural network capable of all possible modes of oscillation. In: Caianiello, E. (ed.): Neural Network. Springer-Verlag (1968) 70–78