The Underlying Formal Model of Algorithmic Lateral Inhibition in Motion Detection

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

[1] Universidad Nacional de Educación a Distancia
E.T.S.I. Informática, 28040 - Madrid, Spain
{jmira,adelgado}@dia.uned.es
[2] Universidad de Castilla-La Mancha
Instituto de Investigación en Informática (I3A) and
Escuela Politécnica Superior de Albacete, 02071 - Albacete, Spain
{caballer,mlopez,miki}@dsi.uclm.es

Abstract. Many researchers have explored the relationship between recurrent neural networks and finite state machines. Finite state machines constitute the best characterized computational model, whereas artificial neural networks have become a very successful tool for modeling and problem solving. Recently, the neurally-inspired algorithmic lateral inhibition (ALI) method and its application to the motion detection task have been introduced. The article shows how to implement the tasks directly related to ALI in motion detection by means of a formal model described as finite state machines. Automata modeling is the first step towards real-time implementation by FPGAs and programming of "intelligent" camera processors.

1 Introduction

Recently the algorithmic lateral inhibition (ALI) method and its application to the motion detection task have been introduced [1]-[5]. And, currently our research team is involved in implementing the method into real-time in order to provide efficient response time in visual surveillance applications [6]-[7].

In recent years, many researchers have explored the relation between discrete-time recurrent neural networks and finite state machines, either by showing their computational equivalence or by training them to perform as finite state recognizers from example [8]. The relationship between discrete-time recurrent neural networks and finite state machines has very deep roots [9]-[11]. The early papers mentioned show the equivalence of these neural networks with threshold linear units, having step-like transfer functions, and some classes of finite state machines. More recently, some researchers have studied the close relationships more in detail [12]-[13], as well as the combination of connectionist and finite state models into hybrid techniques [14]-[15]. During the last decades specialized algorithms even have extracted finite state machines from the dynamics of discrete-time recurrent neural networks [16]-[19].

The article shows how to implement the tasks directly related to ALI in motion detection and introduced by means of a formal model described finite state machines, and the subsequent implementation in hardware, as automata modeling may be considered as the first step towards real-time implementation by field programmable gate arrays (FPGAs) [20] and programming of "intelligent" camera processors.

2 ALI in Motion Detection

The operationalization of the ALI method for the motion detection application has led to the so-called lateral interaction in accumulative computation [2],[4]. From [2],[4] we cite and reformulate the most important concepts and equations.

Temporal Motion Detection firstly covers the need to segment each input image I into a preset group of gray level bands (N), according to equation 1.

$$x_k(i,j;t) = \begin{cases} 1, \text{if } I(i,j;t) \in [\frac{256}{N} \cdot k, \frac{256}{N} \cdot (k+1) - 1] \\ 0, \text{otherwise} \end{cases} \tag{1}$$

This formula assigns pixel (i,j) to gray level band k. Then, the accumulated charge value related to motion detection at each input image pixel is obtained, as shown in formula 2:

$$y_k(i,j;t) = \begin{cases} v_{dis}, \text{if } x_k(i,j;t) = 0 \\ v_{sat}, \text{if } (x_k(i,j;t) = 1) \cap (x_k(i,j;t-\Delta t) = 0) \\ \max[x_k(i,j;t-\Delta t) - v_{dm}, v_{dis}], \\ \qquad \text{if } (x_k(i,j;t) = 1) \cap (x_k(i,j;t-\Delta t) = 1) \end{cases} \tag{2}$$

The charge value at pixel (i,j) is discharged down to v_{dis} when no motion is detected, is saturated to v_{sat} when motion is detected at t, and, is decremented by a value v_{dm} when motion goes on being detected in consecutive intervals t and $t - \Delta t$.

Spatial-Temporal Recharging is thought to reactivate the charge values of those pixels partially loaded (charge different from v_{dis} and v_{sat}) and that are directly or indirectly connected to saturated pixels (whose charge is equal to v_{sat}). Values z_k are initialized to y_k. Formula 3 explains these issues, where v_{rv} is precisely the recharge value.

$$z_k(i,j;t+l\cdot\Delta\tau) = \begin{cases} v_{dis}, & \text{if } z_k(i,j;t+(l-1)\cdot\Delta\tau) = v_{dis} \\ v_{sat}, & \text{if } z_k(i,j;t+(l-1)\cdot\Delta\tau) = v_{sat} \\ \min[z_k(i,j;t+(l-1)\cdot\Delta\tau) + v_{rv}, v_{sat}], \\ \qquad \text{if } v_{dis} < z_k(i,j;t+(l-1)\cdot\Delta\tau) < v_{sat} \end{cases} \tag{3}$$

This step occurs in an iterative way in a different space of time $\tau \ll t$. The value of $\Delta\tau$ will determine the number of times the mean value is calculated.

Spatial-Temporal Homogenization aims in distributing the charge among all connected neighbors holding a minimum charge (greater than v_{dis}) - now, O_k is initialized to z_k. This occurs according to the following equation.

$$\begin{aligned} O_k(i,j;t+m\cdot\Delta\tau) = & \frac{1}{1+\delta_{i-1,j}+\delta_{i+1,j}+\delta_{i,j-1}+\delta_{i,j+1}} \\ & \times[O_k(i,j;t+(m-1)\cdot\Delta\tau)+ \\ & \delta_{i-1,j}\cdot O_k(i-1,j;t+(m-1)\cdot\Delta\tau)+ \\ & \delta_{i+1,j}\cdot O_k(i+1,j;t+(m-1)\cdot\Delta\tau)+ \\ & \delta_{i,j-1}\cdot O_k(i,j-1;t+(m-1)\cdot\Delta\tau)+ \\ & \delta_{i,j+1}\cdot O_k(i,j+1;t+(m-1)\cdot\Delta\tau)] \end{aligned} \tag{4}$$

where

$$\forall(\alpha,\beta)\in[i\pm 1,j\pm 1], \delta_{\alpha,\beta}=\begin{cases}1, & \text{if } O_k(\alpha,\beta;t+(m-1)\cdot\Delta\tau)>v_{dis}\\ 0, & \text{otherwise}\end{cases} \tag{5}$$

Lastly, we take the maximum value of all outputs of the k gray level bands to show the silhouette of a moving object:

$$O(i,j;t)=\arg\max_k O_k(i,j;t) \tag{6}$$

3 Formal Model for ALI in Motion Detection

The control knowledge is described in extensive by means of a finite automaton in which the state space is constituted from the set of distinguishable situations in the state of accumulated charge in a local memory [5]. Thus, we distinguish $N+1$ states $S_0, S_1, ..., S_N$, where S_0 is the state corresponding to the totally discharged local memory (v_{dis}; in general $v_{dis}=0$), S_N is the state of complete charge ($v_{sat}=7$) and the rest are the $N-1$ intermediate charge states between v_{dis} and v_{sat}.

Let us suppose, without loss of generality, that it is enough to distinguish eight levels of accumulated charge ($N=8$) and, consequently, that we can use as a formal model of the control underlying the inferential scheme that describes the data flow corresponding to the calculation of this subtask an 8 states automaton $(S_0, S_1, ..., S_7)$, where S_0 corresponds to v_{dis} and S_7 to v_{sat}. Let us also suppose that discharge ($v_{dm}=2$) and recharge ($v_{rv}=1$) initially take the values corresponding to the descent of two states and to the ascent of one state. This way, the state transition diagram corresponds to a particular kind of reversible counter ("up-down") controlled by the result of the lateral inhibition (dialogue among neighbors).

To complete the description of the states, together with the accumulated charge value, v ($v_{dis}\le v\le v_{sat}$), it is necessary to include come binary signals, A_P and $A_C=0,1$. When $A_P=1$, a pixel tells its neighbors that it has detected a mobile, or that some neighbor has told him to have detected a mobile. A_C indicates that motion has been detected on the pixel.

3.1 ALI Temporal Motion Detecting

The task firstly gets as input data the values of the 256 gray level input pixels and generates $N=8$ binary images, $x_k(i,j;t)$. The output space has a FIFO memory structure with two levels, one for the current value and another one for the previous instant value. Thus, for N bands, there are $2N = 16$ binary values for each input pixel; at each band there is the current value $x_k(i,j;t)$ and the previous value $x_k(i,j;t-\Delta t)$, such that:

$$x_k(i,j;t) = \begin{cases} 1, & \text{if } I(i,j;t) \in [32 \cdot k, 32 \cdot (k+1) - 1] \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Thus, a pair of binarised values at each band, $x_k(i,j;t)$ and $x_k(i,j;t-\Delta t)$, constitutes the input space of the temporal non recurrent ALI. The output space is the result of the individual calculus phase in each calculus element. The inputs are observables $x_k(i,j;t)$ and $x_k(i,j;t-\Delta t)$ and the current charge value that initially is at state S_0. It also receives the comparison rule and the numerical coding of the different discrepancy classes $(D1, D2, D3)$. The output is the class of discrepancy selected at this time, $D(t)$.

$$D(t) = \begin{cases} D2, & \text{if } (x_k(i,j;t) = 1) \cap (x_k(i,j;t-\Delta t) = 0) \\ D3, & \text{if } (x_k(i,j;t) = 1) \cap (x_k(i,j;t-\Delta t) = 1) \\ D1, & \text{otherwise} \end{cases} \tag{8}$$

This class is now an input in charge of filtering a specific charge value (before dialogue) from a set of potential values. These potential values are v_{dis}, v_{sat} and $max[x_k(i,j;t-\Delta t) - v_{dm}, v_{dis}]$. The output of subtask ALI Temporal Motion Detecting constitutes the accumulated charge value, $y_k(i,j;t)$, complemented by label A_C. Remember that $A_C = 1$ denotes the fact that a movement has been locally detected by this pixel.

$$A_C = \begin{cases} 1, & \text{if } D(t) = D2 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$y_k(i,j;t) = \begin{cases} v_{dis}, & \text{if } D(t) = D1 \\ v_{sat}, & \text{if } D(t) = D2 \\ \max[x_k(i,j;t-\Delta t) - v_{dm}, v_{dis}], & \text{if } D(t) = D3 \end{cases} \tag{10}$$

The following situations can be observed in Fig. 1 (see discrepancy class D):

1. $x_k(i,j;t-\Delta t) = 0, 1, x_k(i,j;t) = 0$ (corresponding to discrepancy class $D1$) In this case the calculation element (i,j) has not detected any contrast with respect to the input of a mobile in that band $(x_k(i,j;t) = 0)$. It may have detected it (or not) in the previous interval $(x_k(i,j;t-\Delta t) = 1, x_k(i,j;t) = 0)$. In any case, the element passes to state $S_0[v = v_{dis}, A_C = 0]$, the state of complete discharge, independently of which was the initial state.
2. $x_k(i,j;t-\Delta t) = 0, x_k(i,j;t) = 1$ (corresponding to discrepancy class $D2$) The calculation element has detected in t a contrast in its band $(x_k(i,j;t) = 1)$, and it did not in the previous interval $(x_k(i,j;t-\Delta t) = 0)$. It passes to state $S_7[v = v_{sat}, A_C = 1]$, the state of total charge, independently of which was the previous state. Also A_C passes to 1, in order to tell its potential dialogue neighbors that this pixel has detected a mobile.
3. $x_k(i,j;t-\Delta t) = 1, x_k(i,j;t) = 1$ (corresponding to discrepancy class $D3$). The calculation element has detected the presence of an object in its band $(x_k(i,j;t) = 1)$, and it had also detected it in the previous interval $(x_k(i,j;t-\Delta t) = 1)$. In this case, it diminishes its charge value

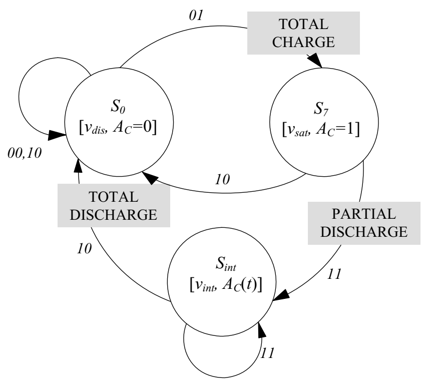

Fig. 1. Control automaton that receives inputs $x_k(i, j; t - \Delta t)$ and $x_k(i, j; t)$, and produces three outputs, coincident with its three distinguishable charge states ($S_0 = v_{dis}$, $S_7 = v_{sat}$, and v_{int})

in a certain value, v_{dm}. This discharge - partial discharge - can proceed from an initial state of saturation $S_7[v_{sat}, A_C = 1]$, or from some intermediate state $(S_6, ..., S_1)$. This partial discharge due to the persistence of the object in that position and in that band, is described by means of a transition from S_7 to an intermediate state, $S_{int}[v_{int}, A_C = 0, 1]$, without arriving to the discharge, $S_0[v_{dis}, A_C = 0]$. The descent in the element's state is equivalent to the descent in the pixel's charge, such that (as you may appreciate on Fig. 1) only the following transitions are allowed: $S_7 \to S_5, S_6 \to S_4, S_5 \to S_3, S_4 \to S_2, S_3 \to S_1, S_2 \to S_0$, and $S_1 \to S_0$.

3.2 ALI Spatial-Temporal Recharging

In the previous task the individual "opinion" of each computation element has been obtained. But, our aim is also to consider the "opinions" of the neighbors. The reason is that an element individually should stop paying attention to motion detected in the past, but before making that decision there has to be a communication in form of lateral inhibition with its neighbors to see if any of them is in state S_7 (v_{sat}, maximum charge). Otherwise, it will be discharging down to S_0 (v_{dis}, minimum charge), because that pixel is not bound to a pixel that has detected motion. The output is formed after dialogue processing with neighboring pixels by the so called permanency value, $z_k(i, j; t)$.

The values of charge accumulated before dialogue are written in the central part of the output space of each pixel (C^*) that now enters in the dialogue phase. The data in the periphery of receptive field in the output space of each pixel (P^*) contains now the individual calculi of the neighbors. Let $v_C(t) = y_k(i, j; t)$ be the initial charge value at this subtask. Each pixel takes into account the set of individual calculus, $v_C(t + k \cdot \Delta\tau), A_j$, by means of the logical union of the labels:

$$A_{P*}(\tau) = \bigcup_j A_j(\tau) \tag{11}$$

This result, A_{P*}, is now compared with A_C, giving rise to one of two discrepancy classes (recharge or stand-by).

$$D(t + l \cdot \Delta\tau) = \begin{cases} stand - by(v_{dis}), \text{ if } v_C(t + l \cdot \Delta\tau) = v_{dis} \\ stand - by(v_{sat}), \text{ if } v_C(t + l \cdot \Delta\tau) = v_{sat} \\ recharge, \text{ if } (v_{dis} < v_C(t + l \cdot \Delta\tau) < v_{sat}) \cap (A_{P*} = 1) \end{cases} \tag{12}$$

Subsequently, the class activated plays the role of selection criteria to output the new consensus charge value after dialogue, $z_k(i, j; t + \Delta t)$, with $\Delta t = k \cdot \Delta\tau$, being k the number of iterations in the dialogue phase, a function of the size of the receptive field. Notice that τ is a parameter that only depends on the size of the objects we want to detect from their motion. So, the purpose of this inference is to fix a minimum object size in each gray level band. The whole dialogue process is executed with clock τ, during k intervals $\Delta\tau$. It starts when clock t detects the configuration $y_k(i, j; t - \Delta t) = y_k(i, j; t) = 1$ and ends at the end of t, when a new image appears.

$$A_C = \begin{cases} 1, \text{ if } D(t + l \cdot \Delta\tau) = \{stand - by(v_{sat}) \cup recharge\} \\ 0, \text{ otherwise} \end{cases} \tag{13}$$

$$v(t + l \cdot \Delta\tau) = \begin{cases} v_{dis}, \text{ if } D(t + l \cdot \Delta\tau) = stand - by(v_{dis}) \\ v_{sat}, \text{ if } D(t + l \cdot \Delta\tau) = stand - by(v_{sat}) \\ \min[v(t + (l-1) \cdot \Delta\tau) + v_{rv}, v_{sat}], \\ \qquad \text{if } (D(t + l \cdot \Delta\tau) = recharge \end{cases} \tag{14}$$

$$A_C = 0, \text{ if } D(t + (l-1) \cdot \Delta\tau) = \{stand - by(v_{sat}) \cup recharge\} \tag{15}$$

In each dialogue phase (in other words, in each interval of clock $\Delta\tau$), the calculation element only takes into account values $y_k(i, j; t - \Delta t)$, $y_k(i, j; t)$ and $A_C(t)$ present in that moment in its receptive field. To diffuse or to use more distant information, new dialogue phases are necessary. That is to say, new inhibitions in $l \cdot \Delta\tau$ $(1 < l \leq k)$ are required. This only affects to state variable $A_C(\tau)$, as $y_k(i, j; t - \Delta t)$ and $y_k(i, j; t)$ values remain constant during the intervals used to diffuse τ and to consensus the different partial results obtained by the calculation elements. Notice that the recharge may only be performed once during the whole dialogue phase. That is why $A_C = 0$, when a recharge takes place. Lastly, the output will be:

$$z_k(i, j; t + \Delta t) = v_C(t + \Delta t) \tag{16}$$

In the corresponding state transition diagram the following situations have to be distinguished:

1. $y_k(i, j; t - \Delta t) = 0, 1, y_k(i, j; t) = 0$
   In any case, independently of the pixel's dialogue with the neighbors, at the end of Δt the pixel passes to state $S_0[v = v_{dis}, A_C = 0]$.
2. $y_k(i, j; t - \Delta t) = 0, y_k(i, j; t) = 1$
   Again, independently of the dialogue phase, the pixel's state will be $S_7[v = v_{sat}, A_C = 1]$.
3. $y_k(i, j; t - \Delta t) = 1, y_k(i, j; t) = 1$
   (a) Local memory is in $S_0[v_{dis}, A_C = 0]$. Pixels in state S_0 are not affected by lateral recharge due motion detection in their periphery. Thus, the pixel maintains the same state S_0.

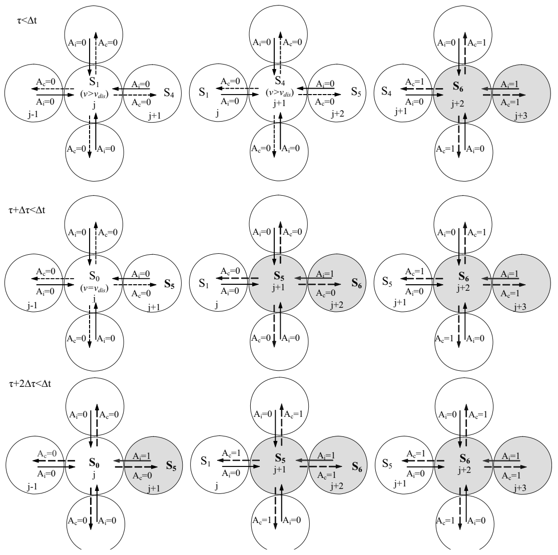

Fig. 2. Detail of the dialogue where diffusion of motion detection is shown through "transparent" pixels ($j+2$ and $j+1$), while pixel j deserves an "opaque" behavior

(b) Local memory is in $S_7[v_{sat}, A_C = 1]$. Pixels in state S_7 are maximally charged. So, they can not be recharged. They also maintain the state.

(c) Local memory is in $S_{int}[v_{int}, A_C(t)]$. Depending on their four neighbors' charge values, it can stay in S_{int} if all neighbors have variable $A_j = 0$ or transit up to S_7 if it finds some neighbor with variable $A_j = 1$.

  i. Transit from S_i to S_{i+1}. After recharge, the calculation element is now in S_{i+1}. It sends $A_C = 1$ and waits up to the end of Δt. In a second clock cycle $\Delta\tau$, $A_C = 1$ is potentially used by its neighbors to increment their charge values. Thus, the dialogue extends in steps of size the receptive field. Pixels with are said to be "transparent" if they allow information on motion detection by some neighbor (in state S_7) of their receptive field to cross them.

  ii. Remain in S_i. If none of its neighbors has transmitted $A_j = 1$, the pixel stays in S_i, without recharging in the first $\Delta\tau$. In this case, it maintains its proper $A_{C^*} = 0$, and its behavior is called "opaque". However, if in a later $\Delta\tau$ and inside the dialogue interval it does receive any $A_j = 1$, it will pass to S_{i+1}. Fig. 2 illustrates this diffusion mechanism through "opaque" and "transparent" pixels of the receptive field.

3.3 ALI Spatial-Temporal Homogenization

Now, the aim of this task is to obtain all moving patches present in the scene. The subtask considers the union of pixels that are physically together and at a

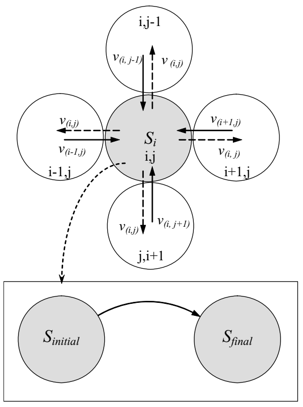

Fig. 3. Dialogue to average the charge values that overcome a threshold inside each gray level band

same gray level band to be a component of an object. A set of recurrent lateral inhibition processes are performed to distribute the charge among all neighbors that possess a certain minimum charge ("permanency value", $z_k(i, j; t)$, of previous task), that is to say, those pixels in states S_1 to S_7, and are physically connected. A double objective is aimed:

1. To dilute the charge due to the image background motion among other points of the own background, so that only moving objects are detected. To dilute the charge due to the image background motion does not mean that we are dealing with moving cameras. Instead of it, we are facing the problem of false motion detected where moving objects are just leaving pixels that now pertain to the background.
2. To obtain a parameter common to all pixels of the object those belong to the same gray level band (simple classification task).

Charge values, $z_k(i, j; t + \Delta t)$ are now evaluated in the center and in the periphery. Now, let v_C be the initial charge value. The result of the individual value (C) is compared with the mean value in (P) - notice that in P^* we have the average of those neighbors that have charge values different from θ_{min}, the so called "permanency threshold value" - and produces a discrepancy class according with threshold, θ_{min}, and passes the mean charge values that overcome that threshold. After this, the result is again compared with a second threshold, namely θ_{max}, eliminating noisy pixels pertaining to non-moving object.

$$D(t + \Delta t) = \begin{cases} D1, \text{ if } v_C = \theta_{min} \\ D2, \text{ if } (\theta_{min} < v_C < v_{sat}) \cap (\theta_{min} < v_P < v_{sat}) \\ D3, \text{ if } (\theta_{min} < v_C < v_{sat}) \cap (v_P = \theta_{min}) \end{cases} \tag{17}$$

$$O_k(i, j; t + \Delta t) = \begin{cases} \theta_{min}, \text{ if } v_C = \theta_{min} \\ (v_C + v_P)/2, \\ \text{ if } (\theta_{min} < v_C < v_{sat}) \cap (\theta_{min} < v_P < v_{sat}) \\ v_C, \text{ if } (\theta_{min} < v_C < v_{sat}) \cap (v_P = \theta_{min}) \end{cases} \tag{18}$$

$$O_k(i, j; t + \Delta t) = v_{dis}, \text{ if } O_k(i, j; t + \Delta t) > \theta_{max} \tag{19}$$

Fig. 3 illustrates the dialogue scheme and the description of the control automaton where the transitions among the initial state $S_i(t)$ (whenever $S_i(t)$ different from S_0) and the final state $S_i(t+\Delta t)$ state are carried out in agreement with rule:

$$S_{i_{final}} = 1/N_{k+1}(S_{i_{initial}} + \sum_{RF_k} v_j) \tag{20}$$

where the sum on sub-index j extends to all neighbors, v_j, belonging to the subset of the receptive field, RF_k, such that its state is different from S_0, and N_k is the number of neighbors with state different from S_0.

4 Data and Results

In order to test the validity of our proposal, in this section the result of applying 8∗8 ALI modules on specific areas of a well-known benchmark image sequences is shown. Figure 4 shows the first and the last images of the Ettlinger-Tor sequence. The treated $64 * 64$-pixel zone of the benchmark is marked.

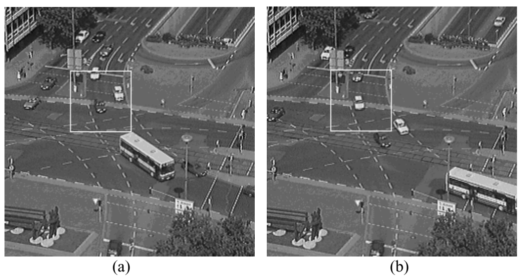

Fig. 4. Ettlinger Tor sequence. (a) Frame number 1. (b) Frame number 40.

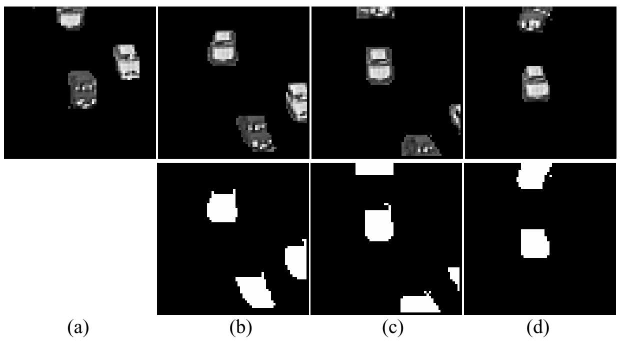

Fig. 5. Ettlinger Tor sequence ground truth and result. (a) After frame number number 1. (b) After frame number 20. (c) After frame number 30. (d) After frame number 40.

Figure 5 shows the result on some frames. After a few frames, the cars are perfectly segmented, as you may appreciate by comparing with the ground truth provided. Again, like in the previous sequence, you may appreciate that there must be enough motion to detect the moving objects. And, concerning the searched real-time performance, let us highlight that the results for $8*8$ modules have been obtained at a frequency of 0.966 MHz (1.035 μs). When extrapolating to usual $512*512$ pixel images, which need 4096 $8*8$ ALI modules, the results should be obtained after 4.24 ms. This performance may be considered as excellent, as in order to work in real-time we have up to 33 ms per image frame.

5 Conclusions

This paper starts from previous works in computer vision, where our algorithmic lateral inhibition method applied to motion detection has proven to be quite efficient. We have shown in this article how the algorithmic lateral inhibition model, based in recurrent neural networks, has been modeled by means of finite state automata, seeking for real-time through an implementation in FPGA-based reconfigurable hardware.

The design by means of programmable logic enables the systematic and efficient crossing from the descriptions of the functional specifications of a sequential system to the equivalent formal description in terms of a Q-states finite state automata or a N-recurrent-neurons neuronal network, where $Q \leq 2^N$. Starting from this point, a hardware implementation by means of programmable logic is very easy to perform. This kind of design is especially interesting in those application domains where the response time is crucial (e.g. monitoring and diagnosing tasks in visual surveillance and security).

Acknowledgments

This work is supported in part by the Spanish CICYT TIN2004-07661-C02-01 and TIN2004-07661-C02-02 grants, and the Junta de Comunidades de Castilla-La Mancha PBI06-0099 grant.

References

1. A. Fernández-Caballero, J. Mira, M.A. Fernández, and M.T. López, Segmentation from motion of non-rigid objects by neuronal lateral interaction, Pattern Recognition Letters, vol. 22, no. 14, pp. 1517-1524, 2001.
2. A. Fernández-Caballero, J. Mira, A.E. Delgado, and M.A. Fernández, Lateral interaction in accumulative computation: A model for motion detection, Neurocomputing, vol. 50C, pp. 341-364, 2003.
3. A. Fernández-Caballero, M.A. Fernández, J. Mira, and A.E. Delgado, Spatio-temporal shape building from image sequences using lateral interaction in accumulative computation, Pattern Recognition, vol. 36, no. 5, pp. 1131-1142, 2003.

4. A. Fernández-Caballero, J. Mira, M.A. Fernández, and A.E. Delgado, On motion detection through a multi-layer neural network architecture, Neural Networks, vol. 16, no. 2, pp. 205-222, 2003.
5. J. Mira, A.E. Delgado, A. Fernández-Caballero, and M.A. Fernández, Knowledge modelling for the motion detection task: The lateral inhibition method, Expert Systems with Applications, vol. 7, no. 2, pp. 169-185, 2004.
6. M.T. López, A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Visual surveillance by dynamic visual attention method, Pattern Recognition, vol. 39, no, 11, pp. 2194-2211, 2006.
7. M.T. López, A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Motion features to enhance scene segmentation in active visual attention, Pattern Recognition Letters, vol. 27, no. 5, pp. 469-478, 2006.
8. R.P. Ñeco, and M.L. Forcada, Asynchronous translations with recurrent neural nets, in Proceedings of the International Conference on Neural Networks, ICNN'97, vol. 4, pp. 2535-2540, 1997.
9. W.S. McCulloch, and W.H. Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, 1943.
10. S.C. Kleene, Representation of events in nerve nets and finite automata, in Automata Studies, Princeton University Press, 1956.
11. M.L. Minsky, Computation: Finite and Infinite Machines, Englewood Cliffs, Prentice-Hall Inc., 1967.
12. R.C. Carrasco, J. Oncina, and M.L. Forcada, Efficient encoding of finite automata in discrete-time recurrent neural networks, in Proceedings of the Ninth International Conference on Artificial Neural Networks, ICANN'99, vol. 2, pp. 673-677, 1999.
13. R.C. Carrasco, and M.L. Forcada, Finite state computation in analog neural networks: Steps towards biologically plausible models, in Lecture Notes in Computer Science, vol. 2036, pp. 482-486, 2001.
14. F. Prat, F. Casacuberta, and M.J. Castro, Machine translation with grammar association: Combining neural networks and finite state models, in Proceedings of the Second Workshop on Natural Language Processing and Neural Networks, pp. 53-60, 2001.
15. G.Z. Sun, C.L. Giles, and H.H. Chen, The neural network pushdown automaton: Architecture, dynamics and training, in Lecture Notes in Artificial Intelligence, vol. 1387, pp. 296-345, 1998.
16. A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland, Finite state automata and simple recurrent networks, Neural Computation, vol. 1, no. 3, pp. 372-381, 1989.
17. C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, and Y.C. Lee, Learning and extracted finite state automata with second-order recurrent neural networks, Neural Computation, vol. 4, no. 3, pp. 393-405, 1992.
18. P. Manolios, and R. Fanelli, First order recurrent neural networks and deterministic finite state automata, Neural Computation, vol. 6, no. 6, pp. 1154-1172, 1994.
19. M. Gori, M. Maggini, E. Martinelli, and G. Soda, Inductive inference from noisy examples using the hybrid finite state filter, IEEE Transactions on Neural Networks, vol. 9, no. 3, p. 571-575, 1998.
20. F. Bensaali, and A. Amira, Accelerating colour space conversion on reconfigurable hardware, Image and Vision Computing, vol. 23, pp. 935-942, 2005.