A parallel algorithm for skeletonizing images by using spiking neural P systems

Daniel Díaz-Pernil [a,*], Francisco Peña-Cantillana [b], Miguel A. Gutiérrez-Naranjo [b]

[a] CATAM Research Group – Department of Applied Mathematics I University of Seville, Spain
[b] Research Group on Natural Computing – Department of Computer Science and AI, University of Seville, Spain

Keywords:

Spiking neural P systems
Skeletonizing
Membrane computing
Guo and Hall algorithm

A B S T R A C T

Skeletonization is a common type of transformation within image analysis. In general, the image *B* is a skeleton of the black and white image *A*, if the image *B* is made of fewer black pixels than the image *A*, it does preserve its topological properties and, in some sense, keeps its *meaning*. In this paper, we aim to use spiking neural P systems (a computational model in the framework of membrane computing) to solve the skeletonization problem. Based on such devices, a parallel software has been implemented within the Graphics Processors Units (GPU) architecture. Some of the possible real-world applications and new lines for future research will be also dealt with in this paper.

1. Introduction

Computer vision [1] is probably one of the most promising challenges for computer scientists in the near future. This growing research area needs contributions from many other scientific fields such as artificial intelligence, pattern recognition, signal processing, neurobiology, psychology or image processing. Computer vision deals with the automated processing of images from the real world in order to extract and interpret pieces of information. Under a computational point of view, a digital image is a function from a two-dimensional surface; each point in the surface is mapped into a set of features such as brightness or colour. The computational treatment of such mappings (or digital images) is the basis of many current applications in computer vision such as optical character recognition (OCR), biometrics, automotive safety or surveillance.

In this paper, we focus on the problem of skeletonizing an image. Skeletonization is one of the approaches for representing a shape with a small amount of information by converting the initial image into a more compact representation and keeping the meaning features. The conversion should remove redundant information, but it should also keep the basic structure. Skeletonization is usually considered as a pre-process in pattern recognition algorithms, but its study is also interesting by itself for the analysis of line-based images as texts, line drawings, human fingerprints or cartography.

Many problems in the processing of digital images have features which make it suitable for techniques inspired by nature. One of them is that the treatment of the image can be parallelized and locally solved. Regardless how large is the image, the process can be performed in parallel in different local areas of it. Another interesting feature is that the local information needed for a pixel transformation can also be easily encoded in the data structures used in natural computing. In the literature, we can find many examples of the use of natural computing techniques for dealing with problems associated to the treatment of digital images. One of the classic examples is the use of cellular automata [2]. Other efforts are related to artificial neural networks as in [3]. In this paper, we use *spiking neural P systems*, a computational model in the framework of membrane computing.

Spiking neural P systems (SN P systems, for short) were introduced in [4] as a new class of distributed and parallel computing devices, inspired by the neurophysiological behavior of neurons sending electrical impulses (*spikes*) along axons to other neurons. SN P systems are the third model of computation in the framework of membrane computing,[1] together with the cell-like model inspired by the compartmental structure and functioning of a living cell and the tissue-like model, based on

* Corresponding author. Tel.: +34 95 455 69 21.
E-mail addresses: sbdani@us.es (D. Díaz-Pernil), frapencan@gmail.com (F. Peña-Cantillana), magutier@us.es (M.A. Gutiérrez-Naranjo).

[1] We refer to [5] for a comprehensive presentation in this area and the P system web page http://ppage.psystems.eu, for the up-to-date information.

intercellular communication and cooperation between cells in a tissue.

Recently, membrane computing techniques have been used for solving problems from digital image. Different P system models have been used for dealing with images, as in [6] where cell-like P systems are used [7–10], where tissue-like P systems are used [11], where the *symmetric dynamic programming stereo* (SDPS) algorithm [12] for stereo matching was implemented by using simple P modules with duplex channels or even [13], where membrane computing and quantum-inspired evolutionary algorithms are combined.

The choice of SN P systems for implementing our bio-inspired solution have some advantages and drawbacks. On the one hand, the use of the flow of spikes among neurons for encoding information allows to use an alphabet with a unique symbol and then, the theoretical model can be translated into a hardware programming language in a natural way. On the other hand, this neurophysiological inspiration has a computational cost, and other bio-inspired implementations obtain better experimental times.

The main contribution of this paper comes from the theoretical side, by presenting the design of a classical skeletonizing algorithm in a novel bio-inspired computational model. Nonetheless, the theoretical advantages of the membrane computing techniques for computer vision need a powerful software and hardware for an effective implementation. In this paper, we present a parallel software developed by using a device architecture called Compute Unified Device Architecture (CUDA^{TM}). It is a general purpose parallel computing architecture that allows the parallel NVIDIA[2] Graphics Processors Units (GPUs) to solve many complex computational problems in a more efficient way than on a Central Processing Unit (CPU). GPUs constitute nowadays a solid alternative for high performance computing, and the advent of CUDA allows programmers a friendly model to accelerate a broad range of applications. The way GPUs exploit parallelism differs from multi-core CPUs, which raises new challenges to take advantage of its computing power. GPU is especially well-suited to address problems that can be expressed as data-parallel computations.

The paper is organized as follows: First, we present the restricted model of SN P systems used in the paper and recall the Guo and Hall algorithm for skeletonizing images. In Section 4, the design of the SN P system for skeletonizing images is presented. Next, we show illustrative examples of the use of our implementation. Finally, Section 6 is dedicated to conclusions and future work.

2. Spiking neural P systems

SN P systems can be viewed as an evolution in membrane computing corresponding to a shift from *cell-* to *neural-like* architectures. In SN P systems, the processing elements are called *neurons* and are placed in the nodes of a directed graph, called the *synapse graph*. The computation is performed by sending electrical impulses among the neurons through the synapses. Such electrical impulses are encoded via a single object type, namely the *spike*, which is placed in the neurons. The number of copies of such object determines the electrical charge of the neuron. Each neuron may also contain rules which allow to send spikes (possibly with a delay) to other neurons, or to remove a given number of spikes from it (*firing* and *forgetting* rules).

Firing rules allow a neuron to send information to other neurons in the form of electrical impulses which are accumulated at the target cell. Forgetting rules remove from the neuron a predefined number of spikes. The application of every rule is determined by checking the contents of the neuron against a regular set associated with the rule. In each time unit, if a neuron can use a rule, then it must be used. If two or more rules could be applied, i.e., there are two or more rules associated to the neuron and the current number of spikes satisfies the conditions of such rules, then only one of them is applied. The rule to be applied is nondeterministically chosen among all the candidates. Thus, the rules are used in the sequential manner in each neuron, but neurons work in parallel with each other. A global clock is assumed, marking the time for the whole system, and hence the functioning of the system is synchronized.

Other biological features have been explored in the framework of SN P systems. One such extension (with mathematical motivation) was introduced in [14], where a neuron can emit more than one spike, if the number of emitted ones is not greater than the consumed ones. Other variants including astrocytes [15], weights, which modify the number of spikes that arrives to a neuron according to the *quality* of the link between neurons [16–18], anti-spikes [19], or neuron division [20] have also been considered. In this paper, we will consider SN P systems with weights in the synapses and rules without delay.[3]

In this way, the restricted model of SN P systems used in this paper can be formally described as follows. A *spiking neural P system* of degree $m \geq 1$ is a construct of the form:

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, in),$$

where

1. $O = \{a\}$ is the singleton alphabet (a is called *spike*);
2. $\sigma_1, \sigma_2, \ldots, \sigma_m$ are *neurons*, of the form $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$, where
   (a) $n_i \geq 0$ is the *initial number of spikes* contained in σ_i;
   (b) R_i is a finite set of *rules* of the following two forms:
      (1) *Firing* rules $E/a^c \rightarrow a^d$, where E is a regular expression[4] over a, and $c \geq d \geq 1$ are integer numbers; if $E = a^b$ (with b an integer number, $b \geq c$), then the rule is usually written in the following simplified form: $a^b/a^c \rightarrow a^d$;
      (2) *Forgetting* rules $a^b/a^c \rightarrow \lambda$, for b and c integer numbers with $b \geq c \geq 1$.
3. $syn \subseteq \{1,2,\ldots,m\} \times \{1,2,\ldots,m\} \times \mathbb{W}$ is the set of synapses between neurons, where the set of weights is $\mathbb{W} = \mathbb{Q} \cap [0,1]$, i.e., the set of rational numbers between 0 and 1. The synapses also verifies that $(i,i,k) \notin syn$ for $1 \leq i \leq m$, $k \in \mathbb{W}$.
4. in is the label of the *input* neuron of Π.

A firing rule $E/a^c \rightarrow a^d \in R_i$ can be applied in neuron σ_i if it contains b spikes, $b \geq c$, and a^b belongs to the language associated to E. For applying rules of type $a^b/a^c \rightarrow a^d$, the neuron must contain exactly b spikes. The execution of these rules removes c spikes from σ_i (thus leaving the remaining spikes in σ_i), and sends d spikes to all the neurons σ_j such that $(i,j,k) \in syn$. The number of spikes that arrives to the neuron j through the synapse (i,j,k) depends on the number d of emitted spikes and the quality of the synapse, encoded by the *weight* k. In each neuron, the number of spikes after a computation step is the number of

[2] http://www.nvidia.com.

[3] For a more general description, see [4].

[4] Along this paper, we use regular expressions of type $a^n a^*$, with $n \in \mathbb{N}$. In this case, the language associated to $a^n a^*$ is the set $\{a^{n+k} | k \in \mathbb{N}\}$. For more details about regular expressions, see, for example, [21].

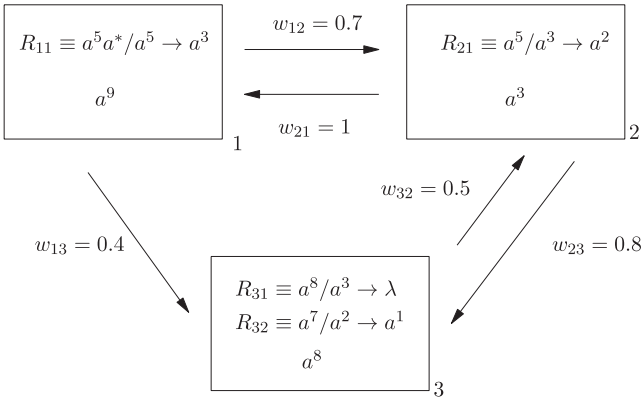


Fig. 1. Example: the input neuron 1 has 9=7+2 spikes at the starting configuration.

non-consumed spikes plus the contribution of other neurons via the corresponding synapses. Let us consider that d spikes are emitted through a synapse (i,j,k). The contribution of σ_i to σ_j is an increase of $\lfloor d \times k \rfloor$ spikes, where $\lfloor v \rfloor$ denotes the largest integer not greater than v. A *forgetting* rule $a^b/a^c \to \lambda$ can be applied in neuron σ_i if it contains *exactly* b spikes. The execution of this rule simply removes all the c spikes from σ_i (thus leaving $b-c$ spikes).

A *configuration* of the system is described by the numbers $\langle n_1,n_2,\ldots,n_m \rangle$ of spikes present in each neuron. At the beginning of the computation, the number of spikes in each neuron σ_i is n_i, but in the input neuron (with label *in*): If the input of the computation is N, then, in the initial configuration, the number of spikes in the input neuron is $n_{in}+N$.

Example 1. Let us consider the SN P system $\Pi = (\{a\},\sigma_1,\sigma_2,\sigma_3,syn,1)$ shown in Fig. 1, with $\sigma_1 = (7,R_1)$, $\sigma_2 = (3,R_2)$, $\sigma_3 = (8,R_3)$ and the set of rules and synapses:

$R_1 = \{R_{11} \equiv a^5a^*/a^5 \to a^3\}$

$R_2 = \{R_{21} \equiv a^5/a^3 \to a^2\}$

$R_3 = \{R_{31} \equiv a^8/a^3 \to \lambda,\ R_{32} \equiv a^7/a^2 \to a^1\}$

$syn = \{(1,2,0.7),(2,1,1),(2,3,0.8),(3,2,0.5),(1,3,0.4)\}$

We will consider the input N=2, so, in order to start the computation, 9 spikes (7+2) are placed in the neuron 1. Notice that R_{11} will be applied if the neuron 1 contains at least 5 spikes. R_{31} is the unique forgetting rule in the SN P system.

Since the input is N=2, the initial configuration is $C_0 = \langle 9,3,8 \rangle$. From this initial configuration, rules R_{11} and R_{31} can be applied. The rule R_{11} consumes 5 spikes from the neuron 1 and sends three spikes to the neurons 2 and 3. These three sent spikes are multiplied by the corresponding weights before arriving to the target neurons. The weight of the synapses between the neurons 1 and 2 is $w_{12} = 0.7$, so the application of the rule R_{11} produces an increase of $\lfloor 3 \times 0.7 \rfloor = 2$ spikes in the neuron 2. Bearing in mind that $w_{13} = 0.4$, the application of the rule R_{11} increases in $\lfloor 3 \times 0.4 \rfloor = 1$ the number of spikes in the neuron 3. The forgetting rule R_{31} deletes three spikes from neuron 3 and hence, the new obtained configuration is $C_1 = \langle 4,5,6 \rangle$.

Now, the unique applicable rule is R_{21}. This rule consumes three spikes from neuron two and sends two spikes to the neurons 1 and 3. According with the corresponding weights, the number of the spikes in the neuron 1 is increased in $\lfloor 2 \times 1 \rfloor = 2$ and the number of the spikes in the neuron 3 is increased in $\lfloor 2 \times 0.8 \rfloor = 1$.

Fig. 2. A hand-written word and its skeletonization. This skeletonization and the ones shown in Figs. 7 and 8 have been obtained by using the CUDA implementation of the corresponding SN P system.

By applying these modifications, the obtained configuration is $C_2 = \langle 6,2,7 \rangle$.

From this configuration, rules R_{11} and R_{32} are applicable. The effects of R_{11} have been described above. The rule R_{32} removes two spikes from the neuron 3 and sends one spike to neuron 2. This spike is multiplied by the corresponding weight, $w_{32} = 0.5$, so it does not produce any increase in the number of spikes in neuron 3, since $\lfloor 0.5 \times 1 \rfloor = 0$. With these changes, the obtained configuration is $C_3 = \langle 1,4,4 \rangle$. No more rules can be applied and C_3 is the halting configuration.

3. Guo and Hall algorithm

Skeletonization is a common transformation in image analysis. The concept of skeleton was introduced by Blum in [22], under the name of medial axis transform. There are many different definitions of the skeleton of a black and white image and many skeletonizing algorithms,[5] but in general, the image B is a skeleton of the image A, if it has fewer black pixels than A, preserves its topological properties and, in some sense, keeps its *meaning*. Fig. 2 illustrates this idea. In this paper, we focus on an iterative procedure of thinning: roughly speaking, the *border* black pixels are removed as long as they are not considered *significant*. The remaining set of black pixels is called the *skeleton*.

Among the parallel algorithms, special attention deserves the so-called 1-subcycle parallel algorithms or fully parallel algorithms [24]. Our bio-inspired design is based on a classical skeletonizing algorithm, the Guo and Hall algorithm [24,25]. In this algorithm, the pixels are examined for deletion in an iterative process.

First of all, given an $p \times q$ image, it is divided into two sub-sections. One of the sections is composed of the pixels a_{ij} such that $i+j$ is even. Alternatively, the second sub-section corresponds to the pixels a_{ij} such that $i+j$ is odd. The algorithm consists of two sub-iterations where the removal of redundant pixels from both sub-sections are alternated, i.e., in each step only the pixels of one of the subsections are evaluated for its deletion.

The decision is based on a 3×3 neighborhood. Given a pixel $P0$, a clockwise enumeration $P1,\ldots,P8$ of its eight neighbor pixels is considered, as shown in Fig. 3(a). As usual, for each $i \in \{1,\ldots,8\}$, Pi is considered as a Boolean variable, with the truth value 1 if Pi is *black* and 0 if Pi is *white*.

In order to decide if a pixel $P0$ is deleted in the corresponding iteration sub-cycle, two parameters are evaluated:

$$B(P0) = \sum_{i=1}^{i=8} Pi$$

[5] A detailed description is out of the scope of this paper. For a survey, see e.g., [23].

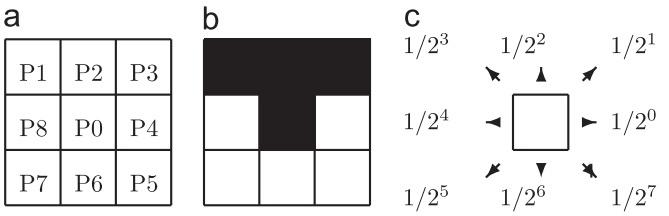


Fig. 3. (a) Enumeration of the pixels in a 3×3 neighborhood; (b) 3×3 neighborhood with encoding [0,0,0,0,1,1,1,1,1], or, shortly, $2^4+2^5+2^6+2^7+2^8=496$ and (c) scheme of the weights of the synapses.

$$C(P0)=(\neg P2\wedge(P3\vee P4))+(\neg P4\wedge(P5\vee P6))+(\neg P6\wedge(P7\vee P8))+(\neg P8\wedge(P1\vee P2))$$

$B(P0)$ counts how many pixels in the neighborhood of $P0$ are black. $C(P0)$ evaluates the *connectivity* of the pixel $P0$. Notice that for isolated black pixels, the connectivity is 0, and for pixels surrounded by eight black pixels, the connectivity is 4.

According to the Guo and Hall algorithm, in each iteration, an evaluated black pixel $P0$ is deleted (changed to white) if and only if all of the following conditions are satisfied.

Guo and Hall conditions:

1. $B(P0)>1$;
2. $C(P0)=1$; This condition is necessary for preserving local connectivity when P is deleted.
3. $(P1\wedge P3\wedge P5\wedge P7)\vee(P2\wedge P4\wedge P6\wedge P8)=FALSE$; Intuitively, this condition is satisfied if $P0$ is not the central pixel of a *cross*.

For example, let us consider as $P0$ the central pixel in the image of Fig. 3(b). In this case, $B(P0)=3>1$, $C(P0)=1$, and the third condition is also satisfied. Hence, $P0$ will be deleted in the corresponding sub-cycle iteration.

4. SN P systems for skeletonizing

In this paper, we will show how to use SN P systems for skeletonizing images. In particular, we use SN P systems for implementing the Guo and Hall algorithm. Without losing generality, we will consider each image as a mapping $I:\{1,\dots,p\}\times\{1,\dots,q\}\to\{black,white\}$, with $I(1,k)=I(p,k)=I(j,1)=I(j,q)=white$ for all $k\in\{1,\dots,q\}$ and $j\in\{1,\dots,p\}$, i.e., the image has $p\times q$ pixels and all the pixels on the border are white.

Given a pixel (i,j), we can use the enumeration of the pixels used in the previous section to represent the neighborhood of the pixel $P0$ in (i,j). Such a neighborhood will be represented as a list $[H0,\dots,H8]$, where for $r\in\{0,\dots,8\}$, $Hr=1$ if Pr is a white pixel and $Hr=0$ if Pr is a black one.[6] This representation of the neighborhood can be done in a more compact way, by encoding the neighborhood as a number[7] in $\{0,\dots,511\}$:

$$cod(i,j)=\sum_{r=0}^{8} Hr\times 2^r$$

For example, in Fig. 3(b), the 3×3 neighborhood can be encoded as [0,0,0,0,1,1,1,1,1], or, shortly, $2^4+2^5+2^6+2^7+2^8=496$.

[6] Notice that we encode black pixels as 0 and white pixels as 1 for an easier implementation with SN P systems. In the previous section, we keep the opposite encoding in order to keep continuity with the literature.

[7] Similar ideas are also used in [23].

Since the decision of removing a black pixel (changing to white) depends on its 3×3 neighborhood and there is a bijective correspondence among the sets of all the possible neighborhoods and the possible encodings $\{0,\dots,511\}$, it is easy to check that the pixel in (i,j) must be removed if it belongs to the set

$$DEL=\left\{\begin{array}{cccccccccccc} 6 & 12 & 14 & 18 & 24 & 26 & 28 & 30 & 36 & 38 & 44 & 46\\ 48 & 50 & 56 & 58 & 60 & 62 & 66 & 72 & 74 & 96 & 98 & 104\\ 106 & 112 & 114 & 120 & 122 & 124 & 126 & 132 & 134 & 140 & 142 & 144\\ 146 & 152 & 154 & 156 & 158 & 164 & 166 & 172 & 174 & 176 & 178 & 184\\ 186 & 188 & 190 & 192 & 194 & 200 & 202 & 224 & 226 & 232 & 234 & 240\\ 242 & 248 & 250 & 252 & 258 & 262 & 264 & 266 & 270 & 286 & 288 & 290\\ 294 & 296 & 298 & 302 & 318 & 384 & 386 & 390 & 392 & 394 & 398 & 414\\ 416 & 418 & 422 & 424 & 426 & 430 & 448 & 450 & 454 & 456 & 458 & 462\\ 480 & 482 & 486 & 488 & 490 & 496 & 498 & 504 & & & & \end{array}\right\}$$

The complementary set of encodings will be denoted by $\overline{DEL}$:

$$\overline{DEL}=\{s\in\{0,\dots,511\}\mid s\notin DEL\}$$

For each image of size $p\times q$, a SN P system will be provided. The input of the SN P system is a non-negative integer which represents the number of iterations in the skeletonizing process. Formally, given an $p\times q$ image, we associate to it the following SN P system is degree $(p\times q)+2$:

$$\Pi=(O,\sigma_{11},\sigma_{12},\dots,\sigma_{pq},\sigma_{odd},\sigma_{even},syn,odd),$$

i.e., a SN P system with a neuron for each pixel in the image plus two extra neurons, σ_{odd} and σ_{even}. The input neuron is σ_{odd}.

- $O=\{a\}$ is the singleton alphabet.
- $\sigma_{odd}=(512,a^{513}a^*/a^{513}\to a^{512})$ and $\sigma_{even}=(0,a^{512}/a^{512}\to a^{512})$.
- $\sigma_{ij}=(n_{ij},R_{ij})$, $i\in\{1,\dots,p\}$, $j\in\{1,\dots,q\}$, where

  $$n_{i,j}=\begin{cases}0 & \text{if } i=1\vee i=p\vee j=1\vee j=q\\ cod(i,j) & \text{otherwise}\end{cases}$$

  $R_{1k}=R_{pk}=R_{j1}=R_{jq}=\emptyset$ for all $k\in\{1,\dots,q\}$ and $j\in\{1,\dots,p\}$. For the remaining (i,j)

  $$R_{ij}=\{a^b/a^{511}\to a^{256}\mid b=r+512\wedge r\in DEL\}\cup\{a^b/a^{512}\to\lambda\mid b=r+512\wedge r\in\overline{DEL}\}$$

- $syn=(\bigcup_{i=2}^{p-1}\bigcup_{j=2}^{q-1} syn_{ij})\cup syn_{odd}\cup syn_{even}\cup\{\langle odd,even,1\rangle,\langle even,odd,1\rangle\}$, where

  $$syn_{odd}=\left\{\langle odd,(i,j),1\rangle \mid \begin{array}{l} i\in\{2,\dots,p-1\},j\in\{2,\dots,q-1\},\\ i+j\ odd\end{array}\right\}$$

  $$syn_{even}=\left\{\langle even,(i,j),1\rangle \mid \begin{array}{l} i\in\{2,\dots,p-1\},j\in\{2,\dots,q-1\},\\ i+j\ even\end{array}\right\}$$

  and for all $i\in\{2,\dots,q-1\}$, $j\in\{2,\dots,q-1\}$,

  $$syn_{ij}=\left\{\begin{array}{ll}\langle(i,j),(i+1,j),1/2^0\rangle, & \langle(i,j),(i-1,j),1/2^4\rangle,\\ \langle(i,j),(i+1,j+1),1/2^1\rangle, & \langle(i,j),(i-1,j-1),1/2^5\rangle,\\ \langle(i,j),(i,j+1),1/2^2\rangle, & \langle(i,j),(i,j-1),1/2^6\rangle,\\ \langle(i,j),(i-1,j+1),1/2^3\rangle, & \langle(i,j),(i+1,j-1),1/2^7\rangle\}\end{array}\right\}$$

The SN P system has one neuron σ_{ij} for each pixel of the image plus two extra neurons σ_{odd} and σ_{even}. The neurons corresponding to the border of the image has zero spikes in the initial configuration, the remaining neurons σ_{ij} corresponding to the pixels

of the image (called hereafter, the *regular neurons*) have $cod(i,j)$ spikes in the initial configuration. The neurons σ_{odd} and σ_{even} have 512 and zero spikes, respectively. We will add to the 512 spikes in σ_{odd} as many spikes as indicated as *input* for starting the computation.

The neuron σ_{odd} has only one rule $a^{513}a^*/a^{513} \to a^{512}$ which is applied if the neuron has at least 513 spikes. The neuron σ_{even} has also one rule $a^{512}/a^{512} \to a^{512}$. The regular neurons have two types of rules: *Firing* and *forgetting* ones, which will be applied if the number of spikes is exactly $b = r+512$ with $r \in DEL$ or $r \in \overline{DEL}$, respectively.

With respect to the synapses, a regular neuron corresponding to a pixel P is linked to the eight neurons corresponding to the eight neighbor pixels of P, with the weights $1/2^i$ where $i \in \{0,\ldots,7\}$ follows an anti-clockwise enumeration of the pixels starting in the *east* pixel, as shown in Fig. 3(c). The neurons σ_{odd} and σ_{even} are linked each other. The neuron σ_{odd} is also linked to all the regular neurons σ_{ij} with $i+j$ odd and, analogously, σ_{even} is linked to all the regular neurons σ_{ij} with $i+j$ even.

4.1. How does it work?

In order to understand how the SN P system works, first we observe that at the initial configuration, the set of regular neurons σ_{ij} encodes the image which will be skeletonized. We will show that, at any time, these neurons encode the successive images obtained in the iterative process of deleting black pixels according to the Guo and Hall algorithm. Let us remark that if the number of spikes in σ_{ij} is even, then the corresponding pixel is black; otherwise, if the number of spikes is odd, then the corresponding pixel is white. This is easily derived from the definition of $cod(i,j)$.

Another observation to be considered is that the parity of the number of spikes in a neuron never changes, since the number of spikes received or removed is always an even amount, except by the application of the rule $a^b/a^{511} \to a^{256}$. As we will see in Lemma 2, the application of this rule is interpreted as the deletion of the corresponding black pixel in the Guo and Hall algorithm and it is applied once at most in each neuron.

Before explaining the different steps of the process, let us consider a pixel (i,j) in the image and a black pixel adjacent to (i,j). We identify this black pixel to $v \in \{P1,\ldots,P8\}$ according to the clockwise enumeration described above. Let us suppose that the black pixel in v belongs to the selected subsection in the current step of the Guo and Hall algorithm and it satisfies the conditions to be deleted.

Let us consider now the neurons σ_{ij} and σ_v corresponding to the pixels in (i,j) and v. As we will show in Lemma 1, the three conditions for v (it is black, it belongs to a selected subsection and it satisfies the Guo and Hall conditions to be deleted) indicate that the number of spikes in σ_v is $b = r+512$ with $r \in DEL$. In this case the rule $a^b/a^{511} \to a^{256}$ is applied. As pointed out above, the application of this rule changes the parity of the number of spikes σ_r (from even, since the pixel is black, to odd) and this change is interpreted as a deletion of the pixel in v.

We focus on the influence of the deletion of the black pixel in v (or, equivalently, the application of the rule $a^b/a^{511} \to a^{256}$ in σ_v) on the neuron σ_{ij}. Since the number of spikes in σ_{ij} at the initial configuration is $cod(i,j) = \sum_{i=0}^{8} Hi \times 2^i$ and, in this configuration, the pixel v is black, according to the encoding, this means that Hv is zero, or, in other words, 2^v does not appear in the encoding of the environment as an addition of powers of 2.

The deletion of the pixel in v changes the environment of (i,j) and then, since the number of spikes in σ_{ij} represents such environment, the number of spikes must change. In particular, the change corresponds to turn Hv to 1, or, in other words, to add 2^v to the number of spikes in σ_{ij}.

In order to check that this happens, it is suffices to see that the rule $a^b/a^{511} \to a^{256}$ sends $256 = 2^8$ spikes from neuron σ_v to σ_{ij}, but these 2^8 spikes must be multiplied by the corresponding weight in $\{1/2^0,\ldots,1/2^7\}$, so only 2, 2^2, 2^3,..., 2^7 or 2^8 spikes arrive to σ_{ij}, depending on the value of v. A simple inspection shows that the number of spikes that arrives to σ_{ij} is exactly 2^v when the black pixel v is deleted.

Bearing in mind these considerations, we show that for an input $N \in \{1,\ldots,513\}$, the computation steps of the SN P system correspond to the iterative process of the Guo and Hall algorithm where the first selected subsection corresponds to pixels with $i+j$ odd. In such way, we will show the following statements:

- **Statement 1:** The set of regular neurons is split into two subsections. One of the sections is composed of the neurons σ_{ij} such that $i+j$ is even. Alternatively, the second sub-section corresponds to the neurons σ_{ij} such that $i+j$ is odd. Both subsections are alternatively selected, starting with the *odd* subsection.
- **Statement 2:** In each computation step, only neurons corresponding to the selected subsection are evaluated. The evaluation consists of determining if the neighborhood of the pixel associated to the neuron satisfies the conditions of the Guo and Hall algorithm to be deleted.
- **Statement 3:** If an evaluated black pixel satisfies the Guo and Hall conditions to be deleted (see Section 3), then the number of spikes in the corresponding neuron changes from *even* to *odd*.
- **Statement 4:** In each configuration, the number of spikes in the regular neurons is the codification of an image, according to the encoding described above.

The first key point of the algorithm is that the image is split into two subsections which will be explored alternatively. One black pixel will be considered for its deletion only if it belongs to the subsection selected in the current step. We consider that a regular neuron σ_{ij} is selected at the step r if its number of spikes in the configuration C_r is greater than or equal to 512. Otherwise, if its number of spikes is lower than 512, then the neuron is not selected.

In the algorithm, the regular neurons with $i+j$ odd and even are alternatively selected. The selection of subsections is performed by the neurons σ_{odd} and σ_{even} which send, alternatively, 512 spikes to the regular neurons σ_{ij} with $i+j$ odd and even, respectively.

Lemma 1. *Let σ_{ij} be a regular neuron and $N \in \{1,\ldots,513\}$ the input of the SN P system. For $r \in \{0,\ldots,N-1\}$*

- *If $i+j$ is odd, then the number of spikes in σ_{ij} is greater than or equal to* 512 *in the configuration C_{2r+1} and it is lower than* 512 *in the configuration C_{2r}.*
- *If $i+j$ is even, then the number of spikes in σ_{ij} is greater than or equal to belongs to*512 *in the configuration C_{2r+2} and it is lower than*511 *in the configuration C_{2r+1}.*

Proof. Let us observe that in the initial configuration, the number of spikes in a regular neuron σ_{ij} is $cod(i,j) < 512$; the number of spikes in σ_{odd} is $512+N$ and there is zero spikes in the neuron σ_{even}. From this initial configuration, the unique applicable rule is $a^{513}a^*/a^{513} \to a^{512}$ in the neuron σ_{odd} (since $N \geq 1$). After applying this rule, in the configuration C_1, the number of spikes in σ_{odd} is $N-1$; the number of spikes in σ_{even} is 512; and the number of spikes in σ_{ij} is $cod(i,j)+512$ if $i+j$ is odd and $cod(i,j)$ if $i+j$ is even.

Let us focus now on σ_{odd} and σ_{even}. The unique neuron that sends spikes to σ_{odd} is σ_{even} and, analogously, the unique neuron that sends spikes to σ_{even} is σ_{odd}. In the configuration C_1, the spikes in σ_{odd} and σ_{even} are $N-1$ and 512, respectively. Since $N \leq 513$, the rule in σ_{odd} cannot be applied in this configuration, but the rule in σ_{even} can be applied, so the spikes in σ_{odd} and σ_{even} in the configuration C_2 are $512+N-1$ and 0, which is similar to the situation in the initial configuration, so we have that for $r \in \{1,\ldots,N\}$, the number of spikes in σ_{odd} and σ_{even} and in the configuration C_{2r} are $512+N-r$ and 0.

Notice that at the configuration C_{2N}, the number of spikes in σ_{odd} and σ_{even} are 512 and 0, respectively, and no more rules are applied in these neurons.

According to the number of spikes in σ_{odd} and σ_{even} in the odd and even configurations, and taken into account their synapses, then we have that, for $r \in \{0,\ldots,N-1\}$, at the configuration C_{2r+1}, the regular neurons with $i+j$ odd has at least 512 spikes; and at C_{2r+2}, the regular neurons σ_{ij} with $i+j$ even has at least 512 spikes.

In order to complete the proof, it is necessary to prove that for $r \in \{0,\ldots,N-1\}$, at the configuration C_{2r+1}, the regular neurons with $i+j$ even has at most 511 spikes; and at C_{2r+2}, the regular neurons σ_{ij} with $i+j$ odd has at most 511 spikes.

Let us start by considering a regular neuron σ_{ij} with $i+j$ odd at the configuration C_1. As we show above, its number of spikes is $b = 512+r$ with $r = cod(i,j) \leq 511$. Depending on $r \in DEL$ or $r \in \overline{DEL}$, one of the rules $a^b/a^{511} \to a^{256}$ or $a^b/a^{512} \to \lambda$ is applied.

- Let us suppose that $r \in DEL$. In particular, this means that the corresponding pixel is black and the applied rule is $a^b/a^{511} \to a^{256}$. The number of spikes in the configuration C_2 is equal to the spikes in the configuration C_1 ($512+r$), minus the consumed ones 511 plus the contribution of other neurons. Since σ_{odd} does not send any spike in this step, the unique contribution to the number of spikes comes from other regular neurons. Each contribution is the addition of 2^i spikes to the spikes in σ_{ij}, but, bearing in mind that the pixel is black, then 2^0 does not appears in the decomposition of $cod(i,j)$ as sum of powers of 2, and it cannot be added as a contribution of other neuron, so r plus the contribution of other neurons is at most 510 and then, the number of spikes in σ_{ij} in C_2 is lower than 512.
- If $r \notin DEL$, then the rule $a^b/a^{512} \to \lambda$ is applied. Since 512 spikes are consumed and r plus the contributions of the other regular neurons is at most 511, then the number of spikes in σ_{ij} is lower than 512, also in this case.

This reasoning can be also applied to show that the number of spikes of the neurons σ_{ij} with $i+j$ even is lower than 512 in the configuration C_3 and in general we have that for $r \in \{0,\ldots,N-1\}$, at the configuration C_{2r+1}, the regular neurons with $i+j$ even has at most 511 spikes; and at C_{2r+2}, the regular neurons σ_{ij} with $i+j$ odd has at most 511 spikes. □

Lemma 1 shows that the property *to have at least* 512 *spikes* changes alternatively from neurons σ_{ij} with $i+j$ odd and even. From this result, it is easy to check the second statement, since the rules in the regular neurons can be only applied if the number of spikes is at least 512. This means that only in such cases the neuron is considered for evaluation.

Evaluating a neuron consists on deciding if the rule $a^b/a^{511} \to a^{256}$ or $a^b/a^{512} \to \lambda$ is applied, but such decision depends on the set *DEL* which are the set of encodings of the neighborhood such that the central pixel must be deleted.

The next key point of the algorithm is the deletion of pixels. By definition of $cod(i,j)$, a regular neuron σ_{ij} has an odd number of spikes if and only if it represents a white pixel. Analogously, a regular neuron σ_{ij} has an odd number of spikes if and only if it represents a white pixel. Bearing in mind this coding of black pixels, deleting a black pixel in a computation step consists on removing an odd amount of spikes from a neuron with an even amount of spikes.

Lemma 2. *Let us consider a black pixel and a step of the Guo and Hall algorithm, such that the pixel belongs to the selected subsection and it satisfies the conditions of the algorithm to be deleted. Then, in the corresponding step of the SN P system computation, the corresponding regular neuron σ_{ij} will pass from an odd amount of spikes to an even amount.*

Proof. According to the previous construction, a black pixel which belongs to the selected subsection in the Guo and Hall algorithm and verifies the conditions to be deleted has associated a regular neuron with $512+r$ spikes, $r \in DEL$. In this case, the rule $a^b/a^{511} \to a^{256}$ is applied. Bearing in mind that r is even, the contributions of other neurons is even and an odd number of spikes is consumed, in the next configuration, the number spikes in the neuron is odd. □

Since all the $r \in DEL$ are even and the contribution of other neurons is always even, then the rule $a^b/a^{511} \to a^{256}$ with $b = 512+r$ is applied at most once in each neuron. This means that if a pixel is deleted (changed from black to white) it never becomes black to white, and the iterative process of thinning the image is also carried out in the SN P system.

Finally, to sum up these statements. We claim the following result.

Theorem. The set of regular neurons of the SN P system encodes in each configuration the successive images obtained in the iterative process of thinning of the Guo and Hall algorithm, by taking as *black* the pixels with an even amount of spikes and *white* the neurons with an odd amount of spikes.

5. Experimental simulation

Simulation of different variants of P systems has been widely studied in the last years. Nonetheless, up to now, no implementation *in vivo* or *in vitro* has been carried out and the natural way to explore the behavior of designed P systems is to simulate it in conventional computers. A short description of some of these simulators can be found in [26,27]. Currently, a big effort is being developed in the *P-lingua project* [28,32], by combining an efficient simulation engine with an *ad-hoc* programming language.

In this paper, a software tool based on the design of the SN P system has been implemented by using Compute Unified Device Architecture (CUDA™) [29]. The experiments have been performed on a computer with a CPU AMD Athlon II × 4 645, which allows to work with four cores of 64 bits to 3.1 GHz. The computer has four blocks of 512 KB of L2 cache memory and 4 GB DDR3 to 1600 MHz of main memory.

The used graphical card (GPU) is an NVIDIA Geforce GT240 composed of 12 *Stream Processors* with a total of 96 cores to 1340 MHz. It has 1 GB DDR3 main memory in a 128 bits bus to 700 MHz. So, the transfer rate obtained is by 54.4 Gbps. The used Constant Memory is 64 KB and the Shared Memory is 16 KB. Its Compute Capability level is 1.2 (from 1.0 to 2.1).

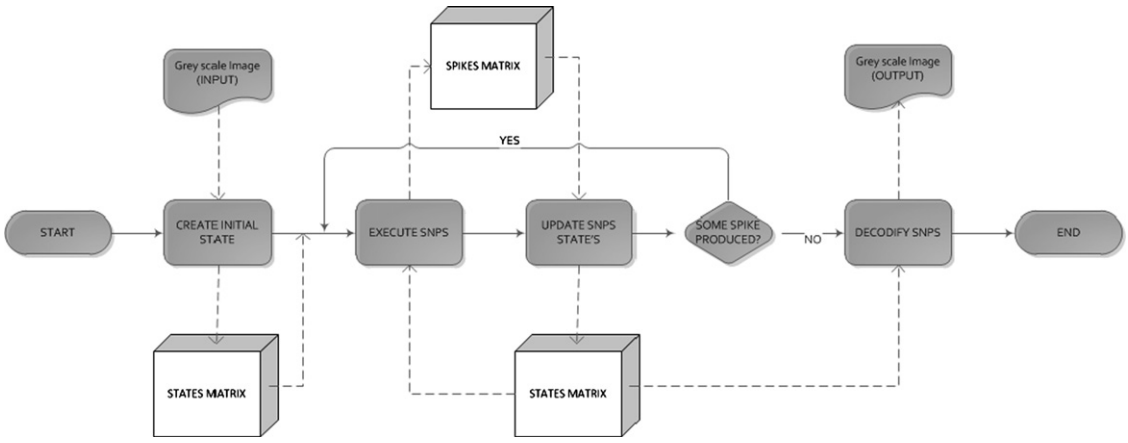


Fig. 4. Workflow of the implementation.

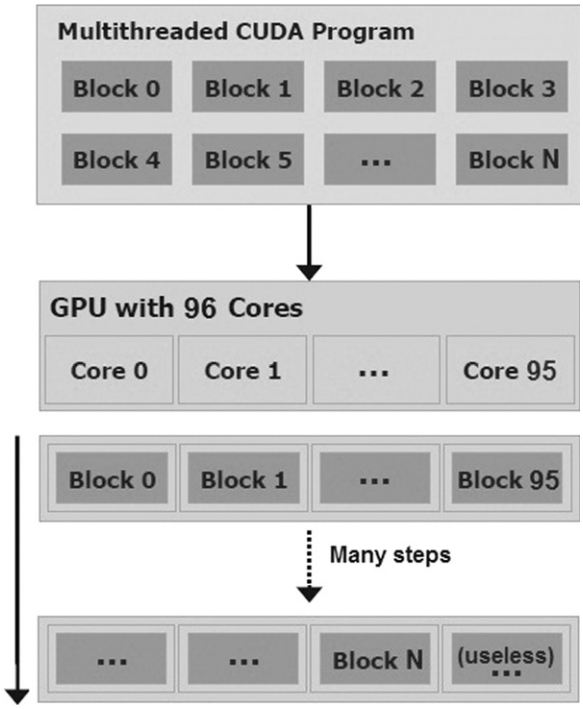


Fig. 5. Scheme of the threads.

In the following, we comment the workflow of our application, where every square in Fig. 4 represents a parallel step using CUDA.

*Step*1: First of all, the `Initial State` is created in parallel, by encoding the pixels from the input image. The matrix is created and stored in the device memory and it is represented as the box STATES MATRIX.

*Step*2: The initial STATES MATRIX is consumed via the execution of the SN P system algorithm. In parallel, all the neurons are checked. If a rule can be applied, then, according to the SN P system model, it is triggered. In order to spread in parallel the number of spikes, four new matrices with the new spikes are created in device memory, called SPIKES MATRIX. The synchronization of the emitted spikes is made via these four matrices.

*Step*3: Next, the state of the SN P system is updated in parallel from the current state and the four SPIKES MATRIX (which are consumed). The next state is stored in STATES MATRIX.

Fig. 6. Example of image with traffic signals.

*Step*4: The neurons are checked again. If the number of the spikes in a neuron has changed (i.e., if a rule has been triggered), then the implementation comes back to Step 2. Otherwise, it goes to Step 5.

*Step*5: The algorithm finishes by creating the output image from the halting configuration of the P system.

The flow of pieces of information between the HOST and the DEVICE is the following:

- The process starts in HOST. The needed information for the parallel calculus is copied into DEVICE. Then, the process that triggers the threads from DEVICE is called.
- The threads are executed in parallel from DEVICE. The threads perform the calculus and the control comes back to the HOST.
- The HOST copies the results from the DEVICE in order to get the information obtained in parallel.

The execution of the threads in parallel depends on the available resources in the GPU: CORES and memory. The execution threads are grouped in 2-dimensional blocks (`Block Threads`). Each block has 16×16 threads. Such Block Threads are grouped in a 2-dimensional grid. For an image of $p \times q$ pixels, at least a grid with $(p/16) \times (q/16)$ Block Threads of 16×16 threads are needed.

The implementation deals with N blocks of threads for the complete image in our GPU of 96 cores. Let us notice if the height and width of the image are not multiples of 16, then more threads

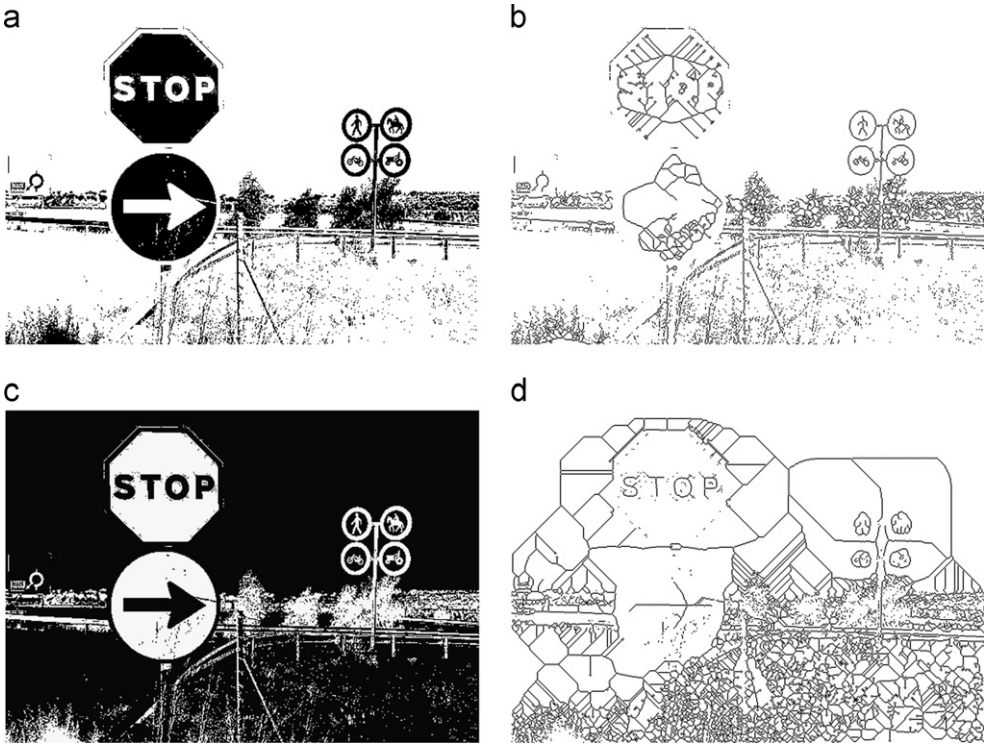


Fig. 7. (a) The binarization of the image from Fig. 6, (b) inverse binarization of the image, (c) its skeletonizing and (d) the skeletonizing of the inverse thresholding.

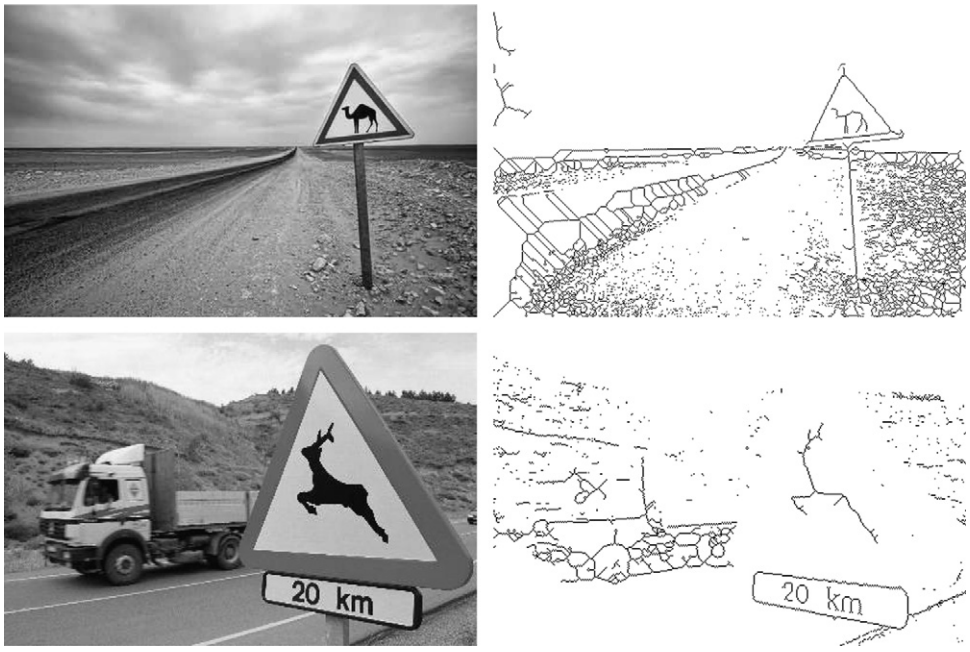


Fig. 8. Original images and their skeletons.

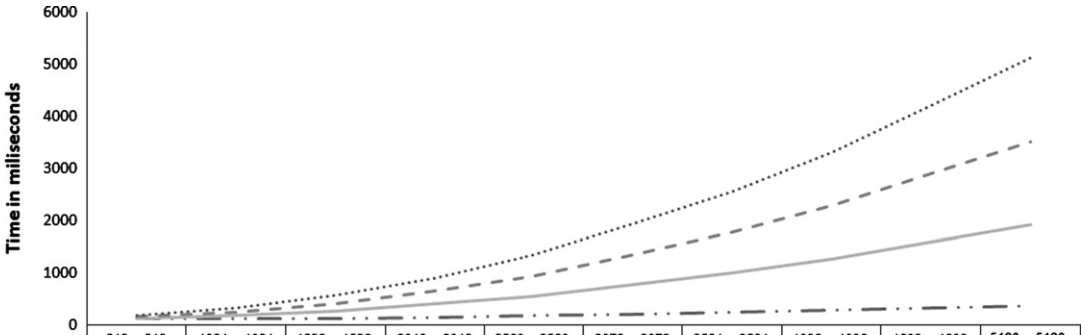
than pixels are needed. In this case, in each block some threads are not used. Fig. 5 shows the sequential steps of the process. In each step the 96 GPU cores are working in parallel. If the number of blocks is not multiple of 96, in the last step there are unused blocks.

5.1. Examples

A first example is shown in Fig. 2. Skeletonizing hand-written texts is one of the challenges of skeletonizing, since the skeleton keeps the topological structure and meaning of the original and the text can be easily stored.[8]

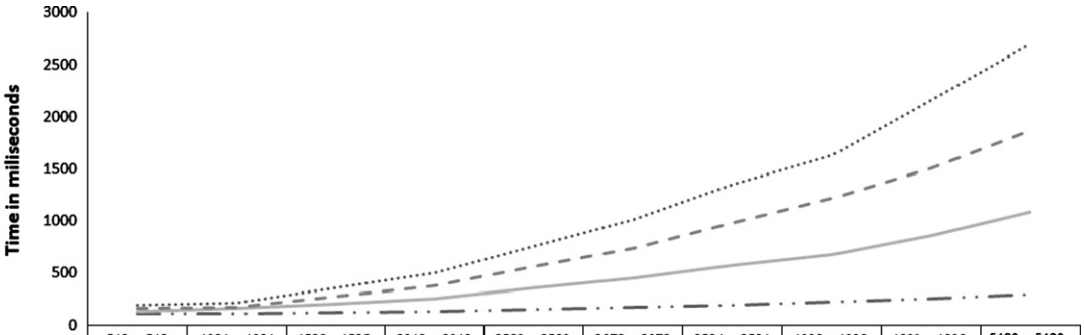
Next, we provide several examples of a realistic recognizing problem with applications in the automotive industry. In Fig. 6, we can see a photograph taken in a road. It has been binarized by using a threshold method by using a threshold 100 on a gray scale 0, . . . ,255. We can see that the skeletonized images in Fig. 7 keep

[8] For a description of the use of skeletonization in image processing, see, for example, [30].



	512 x 512	1024 x 1024	1536 x 1536	2048 x 2048	2560 x 2560	3072 x 3072	3584 x 3584	4096 x 4096	4608 x 4608	5120 x 5120
1 STEP	117	120	133	155	181	212	246	286	334	381
30 STEPS	134	185	275	402	561	772	999	1274	1601	1926
60 STEPS	157	253	422	658	952	1353	1778	2294	2909	3523
90 STEPS	179	321	570	912	1346	1934	2558	3315	4219	5119

Fig. 9. Experimental time obtained for our SN P system implementation of the Guo and Hall algorithm 36 totally black images of $n \times n$ pixels, from $n=512$ to $n=5120$ with a regular increment of 512 pixels of side.



	512 x 512	1024 x 1024	1536 x 1536	2048 x 2048	2560 x 2560	3072 x 3072	3584 x 3584	4096 x4096	4608 x 4608	5120 x 5120
1 STEP	114	115	121	137	155	175	198	224	258	297
30 STEPS	131	168	207	254	358	451	578	675	859	1090
60 STEPS	165	179	274	376	566	739	983	1224	1508	1866
90 STEPS	193	221	360	500	760	1020	1350	1627	2167	2700

Fig. 10. Experimental comparison between the implementation of the Guo and Hall algorithm via Cellular Automata [31] and the implementation via SN P systems presented in this paper.

the information of the traffic signals and they can be used in a further pattern recognition problem. In Fig. 8, two more examples off skeletonizing real images are shown.

We finish this section by showing the results of some experiments performed with our implementation. We have taken 36 totally black images of $n \times n$ pixels,[9] from $n=512$ to $n=5120$ with a regular increment of 512 pixels of side. By taking this initial number of pixel and this increment we are sure that the number of threads for each block is a divisor of the size of the image. Fig. 9 shows the time in milliseconds of our software tool inspired in the designed SN P system for implementing the Guo and Hall algorithm for 1, 30, 60 and 90 steps in the skeletonizing process.

Fig. 10 shows a comparison between the implementation of the Guo and Hall algorithm via Cellular Automata (CA) presented in [31] and the implementation via SN P system shown in this paper. The time of both implementations have been obtained in the same machine from totally black images. The obtained time shows that the current implementation of the SN P system on a GPU takes more time that the CA implementation. One of the reasons for this drawback is that the use of an alphabet with only one object, the spike a, does not fit in the GPU architecture. In this way, further research is needed. In the one hand, new features of the theoretical model can be explored in order to adapt the SN P system to the new hardware architecture. On the other hand, the designed SN P system can be implemented in different hardware architectures by looking for improvements in the efficiency.

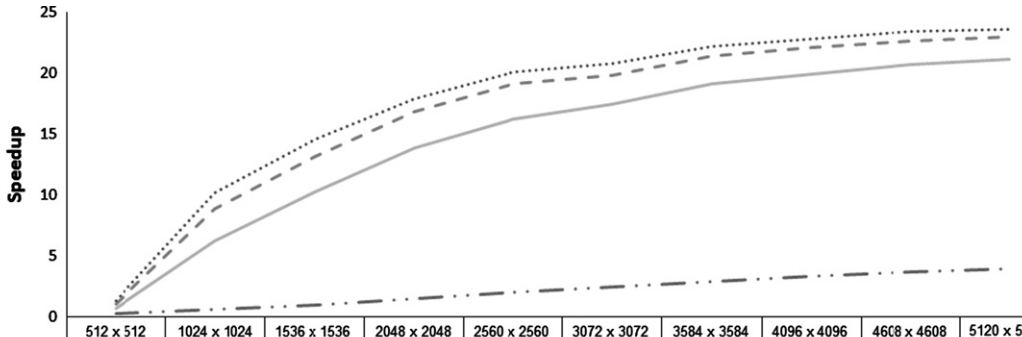
Fig. 11 shows the speed-up of the implementation of the SN P system in the GPU vs. the sequential one, i.e., the ratio of the time taken by both implementations. The study has been made on the same set of black images used in the comparison shown in Fig. 9. As expected, ratio increases when the size of the image or the number of steps increases.

6. Conclusions

The development of new bio-inspired parallel techniques provides a chance for revisiting classical algorithms. In this paper, we have consider a classical algorithm for skeletonizing images, but many other algorithms can be considered. In particular, the bio-inspired computing techniques have features as the encapsulation of the information, a simple representation of the knowledge and parallelism, which are appropriate with dealing with digital images.

The main contribution of this paper is to show that classical parallel algorithms, as the Guo and Hall algorithm, can be implemented in a new computational paradigm inspired in the

[9] Theoretically, this is the worst case, since the time inverted by the algorithm depends on the size of the biggest black connected component of the original image.



	512 x 512	1024 x 1024	1536 x 1536	2048 x 2048	2560 x 2560	3072 x 3072	3584 x 3584	4096 x 4096	4608 x 4608	5120 x 5120
1 STEP	0,29	0,62	1	1,52	2,04	2,49	2,97	3,388	3,72	3,97
30 STEPS	0,75	6,26	10,33	13,85	16,24	17,49	19,12	19,97	20,72	21,17
60 STEPS	1,06	8,87	13,22	16,83	19,15	19,83	21,46	22,09	22,7	23,04
90 STEPS	1,33	10,25	14,58	17,891	20,08	20,78	22,2	22,85	23,44	23,59

Fig. 11. Speed-up of the parallel implementation vs. the sequential one.

communication among neurons. This bio-inspired point of view provides a new perspective on the treatment of digital images. On the one hand, the theoretical model provides decentralized devices, where each neuron acts as an independent processor. On the other hand, the communication among neurons is based on an alphabet with a unique symbol *a*, the spike. This opens a door to future implementations based on electrical impulses.

One of the drawbacks of the used model is that it uses a global clock and the application of rules in the neurons is synchronized. From a biological point of view, it is not realistic that the flow of spikes between any two neurons takes exactly the same time and from the computational point of view, the synchronization of the process has a high cost.

In this paper, we present a novel link between SN P systems and Digital Imagery, by providing an implementation of the parallel Guo and Hall algorithm in this new computational model and we also show how an algorithm designed in this bio-inspired model can be implemented in a natural way in a Graphics Processor Unit, by using its intrinsic features. Nonetheless, the use of new computational paradigms for developing the bio-inspired ideas needs, on the one hand, the contribution of deeper theoretical research that allows us to design new bio-inspired efficient algorithms, and, on the other hand, the use of the most recent parallel computer architectures for a real parallel implementation of the algorithms. In such way, new features can be added to the SN P systems in order to get more efficient designs and other parallel architectures as Field Programmable Gate Arrays (FPGA), grids, ... can be explored for more efficient implementations.

Acknowledgements

DDP and MAGN acknowledge the support of the projects TIN2008-04487-E and TIN-2009-13192 of the Ministerio de Ciencia e Innovación of Spain and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200.

References

[1] L.G. Shapiro, G.C. Stockman, Computer Vision, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
[2] P.L. Rosin, Training cellular automata for image processing, IEEE Trans. Image Process. 15 (2006) 2076–2087.
[3] M. Egmont-Petersen, D. de Ridder, H. Handels, Image processing with neural networks – a review, Pattern Recognition 35 (2002) 2279–2301.
[4] M. Ionescu, G. Păun, T. Yokomori, Spiking neural P systems, Fundam. Inf. 71 (2006) 279–308.
[5] G. Păun, G. Rozenberg, A. Salomaa (Eds.), The Oxford Handbook of Membrane Computing, Oxford University Press, Oxford, England, 2010.
[6] H.A. Christinal, D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, Thresholding of 2D images with cell-like P systems, Rom. J. Inf. Sci. Technol. 13 (2010) 131–140.
[7] H.A. Christinal, D. Díaz-Pernil, P. Real, Segmentation in 2D and 3D image using tissue-like P system, in: E. Bayro-Corrochano, J.-O. Eklundh (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications 14th Ibero American Conference on Pattern Recognition, CIARP 2009, Guadalajara, Jalisco, Mexico, November 15–18, 2009 Proceedings, Lecture Notes in Computer Science, vol. 5856, Springer, Berlin, Heidelberg, 2009, pp. 169–176.
[8] H.A. Christinal, D. Díaz-Pernil, P. Real, Region-based segmentation of 2D and 3D images with tissue-like P systems, Pattern Recognition Lett. 32 (2011) 2206–2212, Advances in Theory and Applications of Pattern Recognition, Image Processing and Computer Vision.
[9] F. Peña-Cantillana, D. Díaz-Pernil, A. Berciano, M.A. Gutiérrez-Naranjo, A parallel implementation of the thresholding problem by using tissue-like P systems, in: P. Real, D. Díaz-Pernil, H. Molina-Abril, A. Berciano, W.G. Kropatsch (Eds.), CAIP (2), Lecture Notes in Computer Science, vol. 6855, Springer, 2011, pp. 277–284.
[10] F. Peña-Cantillana, D. Díaz-Pernil, H.A. Christinal, M.A. Gutiérrez-Naranjo, Implementation on CUDA of the smoothing problem with tissue-like P systems, Int. J. Nat. Comput. Res. 2 (2011) 25–34.
[11] G.Gimel'farb, R. Nicolescu, S. Ragavan, P systems in stereo matching, in: P. Real, D. Díaz-Pernil, H. Molina-Abril, A. Berciano, W. Kropatsch (Eds.), Computer Analysis of Images and Patterns, Lecture Notes in Computer Science, vol. 6855, Springer Berlin/Heidelberg, 2011, pp. 285–292.
[12] G.L. Gimel'farb, Probabilistic regularisation and symmetry in binocular dynamic programming stereo, Pattern Recognition Lett. 23 (2002) 431–442.
[13] G.-X. Zhang, M. Gheorghe, Y. Li, A membrane algorithm with quantum-inspired subalgorithms and its application to image processing, Nat. Comput. (2012) 1–17, http://dx.doi.org/10.1007/s11047-012-9320-2.
[14] H. Chen, M. Ionescu, T.-O. Ishdorj, A. Păun, G. Păun, M. Péz-Jiméz, Spiking neural P systems with extended rules: universality and languages, Nat. Comput. 7 (2008) 147–166, http://dx.doi.org/10.1007/s11047-006-9024-6.
[15] L. Pan, J. Wang, H. Hoogeboom, Spiking neural P systems with astrocytes, Neural Comput. 24 (2012) 805–825.
[16] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, Hebbian learning from spiking neural P systems view, in: D.W. Corne, P. Frisco, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing – 9th International Workshop, WMC 2008, Edinburgh, UK, July 28–31, 2008, Revised Selected and Invited Papers, Lecture Notes in Computer Science, vol. 5391, Springer, Berlin, Heidelberg, 2009, pp. 217–230.
[17] L. Pan, X. Zeng, X. Zhang, Y. Jiang, Spiking neural P systems with weighted synapses, Neural Process. Lett. 35 (2012) 13–27.
[18] J. Wang, H.J. Hoogeboom, L. Pan, G. Păun, M.J. Pérez-Jiménez, Spiking neural P systems with weights, Neural Comput. 22 (2010) 2615–2646.
[19] L. Pan, G. Păun, Spiking neural P systems with anti-spikes, Int. J. Comput. Commun. Control IV (2009) 273–282.
[20] J. Wang, H.J. Hoogeboom, L. Pan, Spiking neural P systems with neuron division, in: M. Gheorghe, T. Hinze, G. Păun, G. Rozenberg, A. Salomaa (Eds.), International Conference on Membrane Computing, Lecture Notes in Computer Science, vol. 6501, Springer, Berlin Heidelberg, 2010, pp. 361–376.
[21] G. Rozenberg, A. Salomaa, Handbook of Formal Languages: Word, language, grammar, Handbook of Formal Languages, Springer, 1997.

[22] H. Blum, An associative machine for dealing with the visual field and some of its biological implications, in: E.E. Bernard, M.R. Kare (Eds.), Biological Prototypes and Synthetic Systems, vol. 1, Plenum Press, New York, 1962, pp. 244–260. (Proceedings of the 2nd Annual Bionics Symposium, held at Cornell University, 1961).
[23] K. Saeed, M. Tabedzki, M. Rybnik, M. Adamski, K3M: a universal algorithm for image skeletonization and a review of thinning techniques, Appl. Math. Comput. Sci. 20 (2010) 317–335.
[24] Z. Guo, R.W. Hall, Parallel thinning with two-subiteration algorithms, Commun. ACM 32 (1989) 359–373.
[25] Z. Guo, R.W. Hall, Fast fully parallel thinning algorithms, CVGIP: Image Understanding 55 (1992) 317–328.
[26] D. Díaz-Pernil, C. Graciani, M.A. Gutiérrez-Naranjo, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Software for P systems, in: G. Păun, G. Rozenberg, A. Salomaa (Eds.), The Oxford Handbook of Membrane Computing, Oxford University Press, Oxford, England, 2010, pp. 437–454.
[27] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Available membrane computing software, in: G. Ciobanu, M.J. Pérez-Jiménez, G. Păun (Eds.), Applications of Membrane Computing, Natural Computing Series, Springer, 2006, pp. 411–436.
[28] D. Díaz-Pernil, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, A P-lingua programming environment for membrane computing, in: D.W. Corne, P. Frisco, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing – 9th International Workshop, WMC 2008, Edinburgh, UK, July 28–31, 2008, Revised Selected and Invited Papers, Lecture Notes in Computer Science, vol. 5391, Springer, Berlin, Heidelberg, 2009, pp. 187–203.
[29] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with CUDA, Queue 6 (2008) 40–53.
[30] K. Ntirogiannis, B. Gatos, I. Pratikakis, An objective evaluation methodology for document image binarization techniques, in: K. Kise, H. Sako (Eds.), Document Analysis Systems, IEEE Computer Society, 2008, pp. 217–224.
[31] F. Peña-Cantillana, A. Berciano, D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, Parallel skeletonizing of digital images by using cellular automata, in: M. Ferri, P. Frosini, C. Landi, A. Cerri, B.D. Fabio (Eds.), CTIC, Lecture Notes in Computer Science, vol. 7309, Springer, 2012, pp. 39–48.
[32] D.W. Corne, P. Frisco, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing – 9th International Workshop, WMC 2008, Edinburgh, UK, July 28–31, 2008, Revised Selected and Invited Papers, Lecture Notes in Computer Science, vol. 5391, Springer, Berlin, Heidelberg, 2009.