Pattern Recognition Letters 34 (2013) 846–855

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Segmenting images with gradient-based edge detection using Membrane Computing

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

[a] CATAM Research Group, Dept. of Applied Mathematics I, University of Seville, Spain
[b] Research Group on Natural Computing, Dept. of Computer Science and AI, University of Seville, Spain
[c] Department of Didactic of Mathematics and Experimental Sciences, University of the Basque Country, Spain

A R T I C L E I N F O

Article history:
Available online 2 November 2012

Keywords:
Edge detection
Sobel algorithm
Tissue P systems
Membrane Computing
CUDA

A B S T R A C T

In this paper, we present a parallel implementation of a new algorithm for segmenting images with gradient-based edge detection by using techniques from Natural Computing. This bio-inspired parallel algorithm has been implemented in a novel device architecture called CUDA™(Compute Unified Device Architecture). The implementation has been designed via tissue P systems on the framework of Membrane Computing. Some examples and experimental results are also presented.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Paralleling classical digital image algorithms is a big challenge for the next years (Parker, 2010; Davies, 2012). Such paralleling is much more complex than the merely simultaneous application of the sequential algorithm to different pieces of the image. The coordination of different simultaneous processes in a whole algorithm is so hard task that commonly the parallel algorithm needs to be re-designed with only slight references to the classical one. Usually, the design of a new parallel implementation not inspired by the sequential one allows an open-mind vision of the problem and the proposal of new creative solutions.

The key point of paralleling classical sequential algorithms is the search of the efficiency and such efficiency is strongly linked to the development of new parallel hardware architectures with allows a realistic implementation of the theoretical advantages of the parallel processes.

In this paper, the matter of study is the Sobel algorithm (Sobel, 1970) for edge detection. We present a parallel implementation of the algorithm in the 3×3 and 5×5 versions. Based on a detailed study of these parallel implementations, in this paper we also introduce a new edge detection algorithm, the so called *AGP segmentator*. A preliminary experimental comparison with the parallel implementation of the 3×3 and 5×5 Sobel operator shows that the *AGP segmentator* improves the classical version of the Sobel operator.

Paralleling classical computer algorithm is currently a vivid research area where different hardware architectures (clusters, grids, FPGA, ...) propose different solutions (Khalid et al., 2011a; Khalid et al., 2011b; Ogawa et al., 2010; Sanduja and Patial, 2012). The chosen hardware architecture for our parallel implementation has been the Compute Unified Device Architecture,[1] CUDA™. This is a novel general purpose parallel computing architecture that allows the parallel NVIDIA Graphics Processing Units (GPUs) to solve many complex computational problems in a more efficient way than on a CPU. The choice of this parallel architecture is supported by several reasons. The first one is that the computing language CUDA™ allows programmers a friendly model for implementing easily parallel programs, but the main reason comes from the practical side. In the last years, there exists an increasing interest in the specialized industry for the development of more and more powerful Graphic Processing Units which can be used for general purposes. This interest leads, on the one hand, to a more economically accessible (and hence, more extended) hardware and, on the other hand, to the development of more powerful computational units.

The design of new parallel solutions needs a strong theoretical support that allows to control, to formalize, to check and even, sometimes, to formally verify new algorithms. As a novel contribution with respect to recent contributions found in the literature, the theoretical foundation of our parallel implementation of the

* Corresponding author.
E-mail addresses: sbdani@us.es (D. Díaz-Pernil), ainhoa.berciano@ehu.es (A. Berciano), frapencan@gmail.com (F. Peña-Cantillana), magutier@us.es (M.A. Gutiérrez-Naranjo).

[1] See http://www.nvidia.com/object/cuda_home_new.html.

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.10.014

Edge Detection algorithms is based on Natural Computing processes, namely, on Membrane Computing techniques.

As it will be shown below, Membrane Computing techniques are inspired in the flow of metabolites between cells of a living tissue or between the organelles in an eucaryotic cell. This flow of metabolites takes place in parallel in Nature and can be interpreted as a flow of information for computational purposes. Instead of a set of few instructions with complex data structures, the computation steps in a Membrane Computing device are regulated by a set of rules with a notation close to biochemical reactions. From a computational point of view, such reactions can be read as a set of *if A then B* rules where A and B are very simple data. As we will show below, this theoretical construction fits perfectly for a computational implementation within the GPU architecture.

The paper is organized as follows: firstly, we recall some preliminaries on Natural Computing and the definition of the used model of tissue P systems. Next, we provide a short description of the algorithms object of our study: thresholding and the Sobel algorithm for edge detection. In Section 4 some details of the implementation and several examples are provided. Finally, some remarks are given in the last section.

2. Natural computing

Nature is a big source of inspiration for new computational paradigms. Nature *acts* by performing changes (from microscopic biochemical reactions to ecological global variations) which can be interpreted as *computations*. Natural Computing[2] abstracts the way Nature operates, providing ideas for new computing models. It involves research where the physical support is non standard, as *DNA-based Molecular Computing* (Adleman, 1994) or *Quantum Computing* (Hirvensalo, 2004); but almost all the research lines in Natural Computing are currently supported in silicon-based computers. Among them, we can cite *Artificial Neural Networks* (McCulloch and Pitts, 1943), *Genetic Algorithms* (Holland, 1992), *Swarm Intelligence* (Engelbrecht, 2005), *Artificial Immune Systems* (de and Timmis, 2002), *Amorphous Computing* (Abelson et al., 2000), *Membrane Computing* (Păun, 2002) or *Cellular Automata* (von Neumann, 1966).

All these computational paradigms have in common the use of an alternative way of encoding the information and the use of intrinsic parallelism of natural processes. In this paper, we will use the theoretical framework of Membrane Computing for handling digital images. The use of techniques inspired in Nature for processing digital images is not new. Many problems in such processing have features which make it suitable for techniques inspired by nature. One of them is the treatment of the image can be parallelized and locally solved. Regardless how large is the picture, the segmentation process can be performed in parallel in different local areas of the picture. 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 (Rosin, 2006; Selvapeter and Hordijk, 2009). Other efforts are related to Artificial Neural Networks (Egmont-Petersen et al., 2002).

In Membrane Computing, there is a large tradition in the study of dealing with information structured as two dimensional objects (see, e.g., (Ceterchi et al., 2003a; Ceterchi et al., 2003b; Dersanambika and Krithivasan, 2004; Krishna et al., 2001)). The main motivation for these studies is to bring together Membrane Computing and Picture Grammars. Recently, a new research line has been opened by applying well-known Membrane Computing techniques for solving problems from Digital Imagery as *segmentation* (Christinal et al., 2009; Christinal et al., 2011; Díaz-Pernil et al., 2010b; Díaz-Pernil et al., 2011), *thresholding* (Christinal et al., 2010a), *smoothing* (Peña-Cantillana et al., 2011b) or the *symmetric dynamic programming stereo* algorithm (Gimel'farb et al., 2011).

The theoretical model used in this paper, Membrane Computing, is a model of computation inspired by the structure and functioning of cells as living organisms able to process and generate information. In particular, it focusses on membranes, which are involved in many reactions taking place inside various compartments of a cell. They act as selective channels of communication between different compartments as well as between the cell and its environment (Alberts et al., 2002). The computational devices in Membrane Computing are called *P systems* (Păun, 2000). Roughly speaking, a P system consists of a membrane structure, in whose compartments one places multisets of objects which evolve according to given rules which are usually applied in a synchronous non-deterministic maximally parallel manner.[3] We stress here on the so-called (because of their membrane structure) *tissue P Systems* (Martín-Vide et al., 2003) endowed with cell division.

2.1. Tissue P systems with cell division

In this section we present the formal bio-inspired model where we have implemented our edge detection algorithms. First of all, let us recall some basic preliminaries.

An *alphabet*, Σ, is a non empty set, whose elements are called *symbols*. An ordered sequence of symbols is a *string*. The number of symbols in a string u is the *length* of the string, and it is denoted by $|u|$. As usual, the empty string (with length 0) will be denoted by λ. A *multiset* m over a set A is a pair (A,f) where $f : A \to \mathbb{N}$ is a mapping. If $m = (A,f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in A | f(x) > 0\}$ and its *size* is defined as $\sum_{x\in A} f(x)$. A multiset is empty (resp. finite) if its support is the empty set (resp. finite). If $m = (A,f)$ is a finite multiset over A, and $supp(m) = \{a_1,\dots,a_k\}$, then it will be denoted as $m = \{\{a_1^{f(a_1)},\dots,a_k^{f(a_k)}\}\}$. That is, superscripts indicate the multiplicity of each element, and if $f(x) = 0$ for any $x \in A$, then this element is omitted. A *graph* G is a pair $G = (V,E)$ where V is the set of vertices and E is the set of edges, each one of which is a (unordered) pair of (different) vertices. In what follows we assume the reader is already familiar with the basic notions and the terminology underlying P systems.

Tissue P systems with cell division is a well-established P system model presented by Păun et al. in (Păun et al., 2008). The biological inspiration for considering *cell division* in this model is that alive tissues are not *static* network of cells, since cells are duplicated via mitosis in a natural way. Tissue P systems with cell division have been previously used to design solutions to **NP**-complete problems in polynomial time (see (Díaz-Pernil et al., 2007; Díaz-Pernil et al., 2008a) and the references therein).

Formally, a *tissue P system with cell division* of degree $q \geqslant 1$ is a tuple of the form

$$\Pi = (\Gamma, \Sigma, \mathcal{E}, w_1, \dots, w_q, \mathcal{R}, i_\Pi, i_0),$$

where:

- Γ is a finite *alphabet*, whose symbols will be called *objects*;
- $\Sigma (\subset \Gamma)$ is the input alphabet;

[2] An introduction on Natural Computing can be found in (de Castro, 2007; Kari and Rozenberg, 2008).

[3] We refer to (Păun, 2002) for basic information in this area, to (Păun et al., 2010) for a comprehensive presentation and the P system web page http://ppage.psystems.eu, for the up-to-date information.