Parallel Skeletonizing of Digital Images by Using Cellular Automata

Francisco Peña-Cantillana[1], Ainhoa Berciano[2,3],
Daniel Díaz-Pernil[3], and Miguel A. Gutiérrez-Naranjo[1]

[1] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Seville, Spain
frapencan@gmail.com, magutier@us.es
[2] Departamento de Didáctica de la Matemática y de las Ciencias Experimentales
University of the Basque Country
ainhoa.berciano@ehu.es
[3] Research Group on Computational Topology and Applied Mathematics
Department of Applied Mathematics
University of Seville, Spain
sbdani@us.es

Abstract. Recent developments of computer architectures together with alternative formal descriptions provide new challenges in the study of digital Images. In this paper we present a new implementation of the Guo & Hall algorithm [8] for skeletonizing images based on Cellular Automata. The implementation is performed in a real-time parallel way by using the GPU architecture. We show also some experiments of skeletonizing traffic signals which illustrates its possible use in real life problems.

1 Introduction

A car at 120 km/h takes 6 seconds to travel 200 meters. Recognizing traffic signals in real time is a challenge in the automotive industry and represents an important step for automatic driving [13,14]. Moreover, changes in visibility due to the weather, lighting etc. make necessary to store an *scheme* of the signal, instead of a picture of it. The *skeleton* of the signal can be a good way to represent the signal in an abstract way. Skeletonizing such images efficiently is crucial in the process, since a car automatically driven must react in a very short space of time according to signals.

In this paper we propose a real-time parallel implementation of the Guo & Hall algorithm [8] for skeletonizing images by using a device architecture called CUDA™, (Compute Unified Device Architecture) [21]. CUDA™ is a general purpose parallel computing architecture that allows the parallel NVIDIA Graphics Processors Units (GPUs) to solve many complex computational problems in a more efficient way than on a CPU [15].

The algorithm has been adapted on the theoretical basis using one of the most known models of the Natural Computing, *Cellular Automata* (CA). Natural Computing studies computational paradigms inspired from physics, chemistry and biology [11]. It abstracts the way in which nature acts, providing ideas for new computing models. One of the main research lines in Natural Computing is Cellular Automata, introduced by John Von Neumann with the biological motivation of obtaining self-replicating artificial systems that are also computationally universal.

CA are decentralized discrete computational systems[1]. They consist of large numbers of simple identical components (cells) placed on an N-dimensional grid with local connectivity defining the *neighbourhood* of a cell. The cells are in one of a finite set of *states*. A discrete global clock is assumed and the cells change their states synchronously depending on their own state and the states of the neighbours, as determined by a local update rule. Technically, a CA consists of two components. The first one is a *cellular space*: a lattice of N identical finite-state machines (cells) each with an identical pattern of local connections to other cells, with boundary conditions if the lattice is finite. The second component is a set of transition rules that gives the update state of each cell.

These features make CA suitable for dealing with some problems in the analysis of digital images, where pixels are identified with cells and the changes in a cell depends on the current state plus the current state of its neighbours and all the changes can be made simultaneously [9,17]. Nonetheless, the inherent features of CA for dealing with Image Analysis have found the limits of sequential computers. The theoretical parallel framework of CA could not be efficiently implemented in one-processor computers. Recently, the development of new parallel architectures has brought a renewed interest in the use of CA for Image Analysis.

The paper is organised as follows: Firstly, we recall the basic notions of the Guo & Hall algorithm and give some ideas of our implementation on CUDA inspired on CA. In Section 3 we show illustrative examples of the use of our implementation and some comparisons with a sequential one. Finally, Section 4 is dedicated to conclusions and future work.

2 Guo and Hall Algorithm

Representing a shape with a small amount of information is a challenge in computer vision. Skeletonization is one of the approaches to this purpose, 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.

[1] We assume that the reader is familiar with the basic concepts of CA. More information in [10,20].

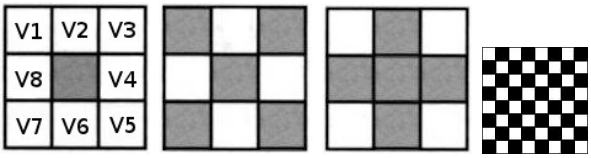

Fig. 1. (a) Order of neighbour pixels of P. (b) Sail configuration. (c) Cross configuration. (d) Chess configuration of an image.

The concept was introduced by Blum [5,6] under the name of medial axis transform. There are many algorithms published in this topic (see [16]) and there are many different approaches to the problem, among them the ones based on distance transform of the shape and skeleton pruning based on branch analysis[2].

In this paper, the skeleton is obtained by an iterative procedure of thinning: the border points are removed as long as they are not considered significant. The remaining set of points is called the *skeleton*. Among the parallel algorithms following this idea, special attention deserves the so-called 1-subcycle parallel algorithms or fully parallel algorithms [8]. We present a new implementation of the algorithm of Guo & Hall by using the new technology GPGPU. In this algorithm, the contour pixels are examined for deletion in an iterative process. The decision is based on a 3×3 neighbourhood. The image is divided into two disjoint areas (sub-sections), similarly to a chess board. The algorithm consists on two sub-iterations where the removal of redundant pixels from *white* and *black* sections are alternated. This is repeated until there are no redundant pixels left.

According to [7], given a pixel P, we will denote by $P_1, \ldots, P_8$ the clockwise enumeration of its eight neighbour pixels (see Fig. 1 a)) and P as a Boolean variable, with the truth value 1 if P is *black* and 0 if P is *white*. Two parameters are defined:

$$\begin{aligned} B(P) &= \textstyle\sum_{i=1}^{i=8} P_i \\ C(P) &= (\neg P_2 \wedge (P_3 \vee P_4)) + (\neg P_4 \wedge (P_5 \vee P_6)) \\ &\quad +(\neg P_6 \wedge (P_7 \vee P_8)) + (\neg P_8 \wedge (P_1 \vee P_2)) \end{aligned}$$

where $B(P)$ is the number of black neighbour pixels, and $C(P)$ is the *connectivity operator* given by the number of white neighbour pixels of P where some of the next two pixels is black, following the order of pixels of Fig. 1 (a).

In each iteration, each pixel P is deleted (changed to white) if and only if all of the following conditions are satisfied:

1. $C(P) = 1$; this condition is necessary for preserving local connectivity when P is deleted.
2. $(P_1 \wedge P_3 \wedge P_5 \wedge P_7) \vee (P_2 \wedge P_4 \wedge P_6 \wedge P_8) = FALSE$; i.e., we cannot to find a configuration of neighbour pixels with the way of Fig. 1 b) and c)).
3. $B(P) > 1$.

[2] See, for example, [1,3,19,4,2].

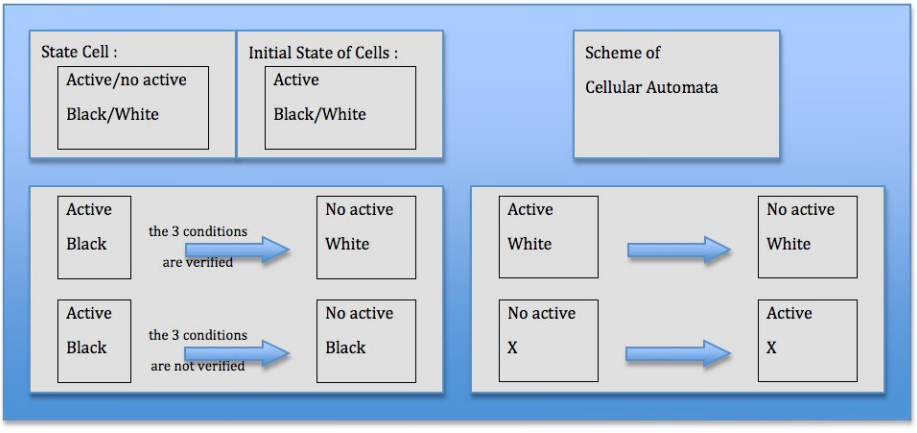

Fig. 2. Scheme of our Cellular Automata

2.1 CA Formal Framework

The formal description of the algorithm in the framework of CA requires to provide the cellular space and the set of rules.

- Given a $n \times m$ image, let us consider the CA *cellular space* on a rectangular grid $(n+2) \times (m+2)$ with 8-adjacency[3]. The set of possible states is $\{x, y\}$ for $x \in \{0, 1\}$, where 0 stands for white and 1 stands for black, and $y \in 0, 1$ where 0 stands for active and 1 for no active. In the initial configuration, we will consider the central $n \times m$ grid, where the cells will take the initial state according with the color of the corresponding pixel in the image. The remaining one-width framework will have the initial state 0 (white). The initial value of y depends of the position of pixel in a image with the same size of the input image, but taking a chess configuration (see Fig. 1 (d)). So, if the pixel is black we consider the cell as active and if the pixel is white the cell is no active.
- For the description of the set of rules, it is necessary to provide the conditions necessary for the change in the state of a pixel. In this case, such conditions are taken directly from the Guo & Hall algorithm (see above). A cell will change its state if we can apply one of the four rules that appear in Fig. 2; for example, a cell changes its state from $\{1, 1\}$ to $\{0, 0\}$ (from {black,active} to {white,no active}) if it satisfies the conditions to be considered *redundant*, which includes its current state and the color of the pixels in its 8-neighbourhood.

[3] We consider this extra framework in order to avoid boundary conditions in the description of the algorithm.

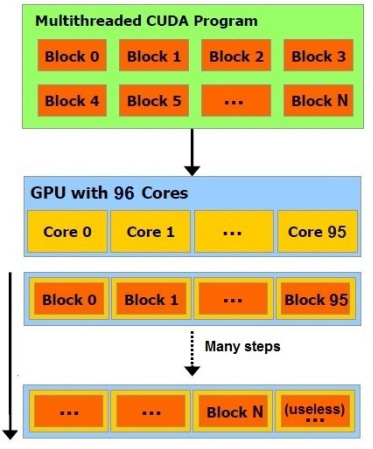

Fig. 3. Scheme of threads for our skeletonizing

2.2 Parallel Implementation

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 parallel implementation of the Guo & Hall algorithm described above has been developed by using Microsoft Visual Studio 2008 Professional Edition (C++) with the plugging Parallel Nsight (CUDA™) under Microsoft Windows 7 Professional with 32 bits. CUDA™ C, an extension of C for implementations of executable kernels in parallel with graphical cards NVIDIA has been used to implement the CA. It has been necessary the *nvcc compiler* of CUDA™ Toolkit and some libraries from openCV to the treatment of input and output images.

The experiments have been performed on a computer with a CPU AMD Athlon II x4 645, which allows to work with four cores of 64 bits to 3.1 GHz. The computer has four blocks of 512KB 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 by 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).

We can deal N blocks of threads for the complete image in our GPU of 96 cores, as we can see in Fig. 3. We need more threads than pixels if the height and width of the image are not multiples of 16, i.e., we can have useless threads. The Figure 4 shows a flowchart of the implementation on CUDA of the algorithm presented in this paper.

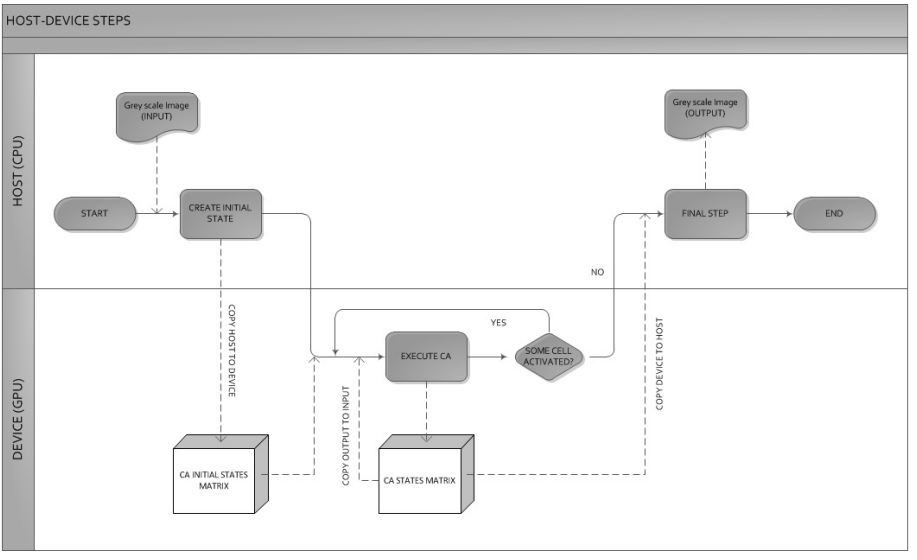

Fig. 4. Flowchart

3 Skeletonizing

In this section we show the results of some experiments of skeletonizing traffic signals with our parallel implementation of the Guo & Hall algorithm. Notice that, the algorithm is described for black and white colour images, so if the image given is a colour image or an image in grey scale, firstly we apply an algorithm of binarization, and later our parallel software.

Firstly, in Figure 5 we can see an example of skeletonizing some traffic signals. Notice that the skeletonizing of signals with the original foreground in black and the background white provides meaningful information about the message, but in the opposite way, the output of the skeletonizing is not a recognisable image and not suitable for automatic recognition.

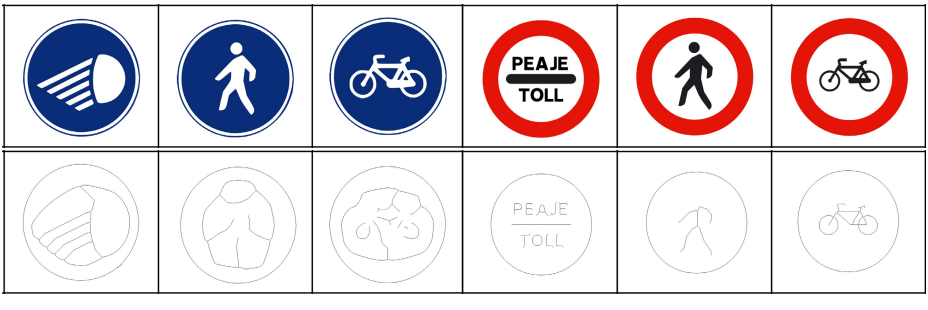

Fig. 5. Some traffic signals and their skeletonization

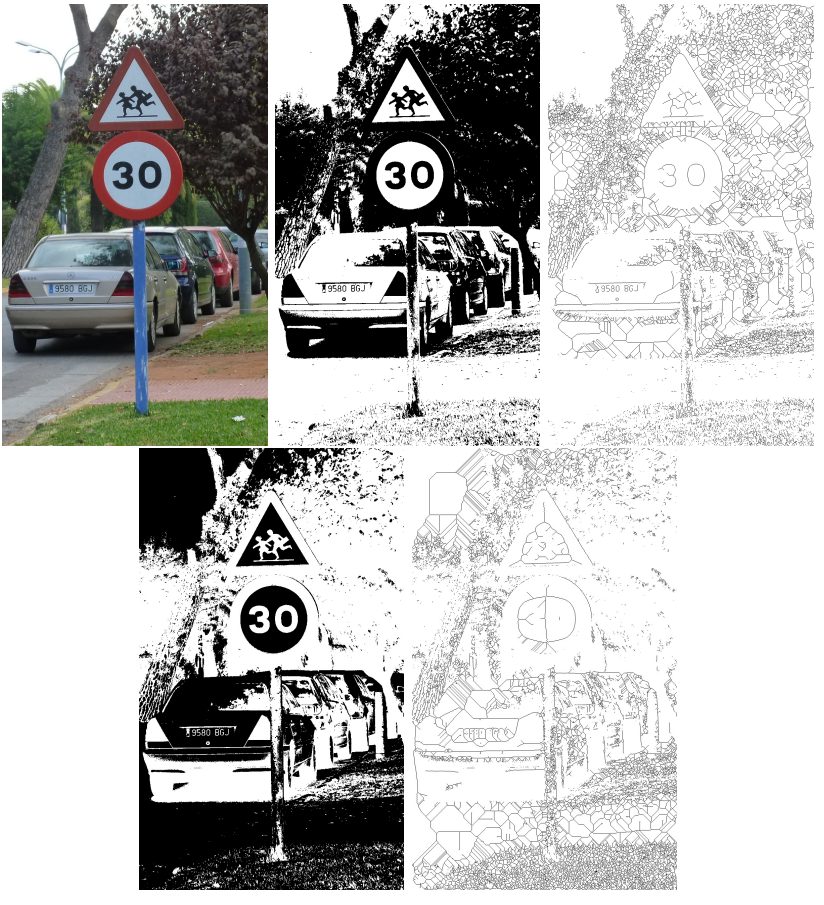

Fig. 6. A real photograph, its binarization, its skeletonizing, its inverse binarization and its inverse skeletonizing (from left to right)

Next, we provide an example of a realistic recognising problem. In Figure 6, we can see a photograph of size 789×1317. It has been binarized by using a threshold method by using a threshold 100 on a gray scale $0, \ldots, 255$. At the bottom of the Figure, we can see its skeletonizing and the skeletonizing of the inverse thresholding.

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[4], from $n = 125$ to $n = 4500$ with a regular increment of 125 pixels of side. Figure 7 (top) shows the time in milliseconds of the application of our implementation of the Guo & Hall algorithm in CA for 1, 30, 60 and 90 steps in the skeletonizing

[4] 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.

process. Figure 7 (bottom) shows the same study for a sequential implementation of the algorithm. Finally, Figure 8 shows a comparison of our implementation vs. the sequential one by taking 90 steps in the Guo & Hall algorithm.

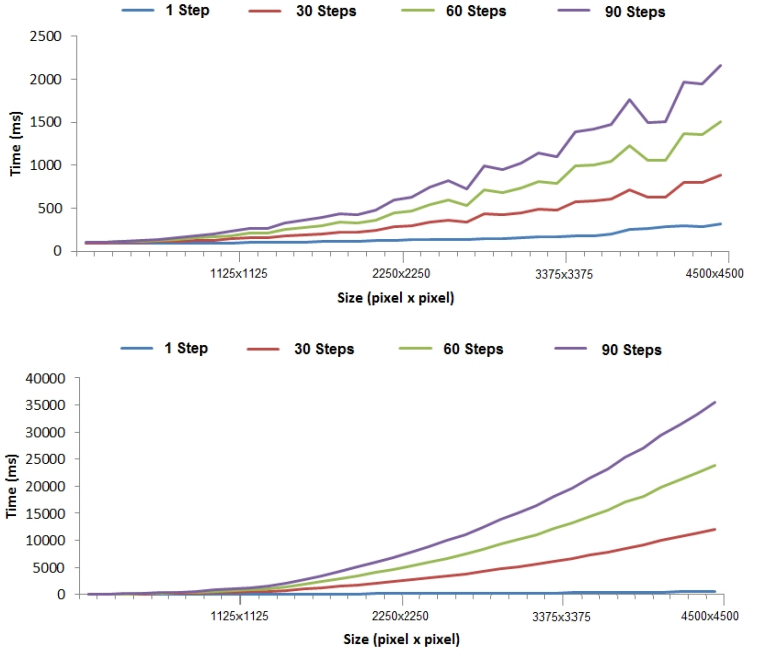

Fig. 7. Experimental time obtained for the Guo & Hall algorithm 36 totally black images of $n \times n$ pixels, from $n = 125$ to $n = 4500$ with a regular increment of 125 pixels of side. Top image shows the time of our parallel implementation in CA. Bottom image shows the time for a sequential implementation.

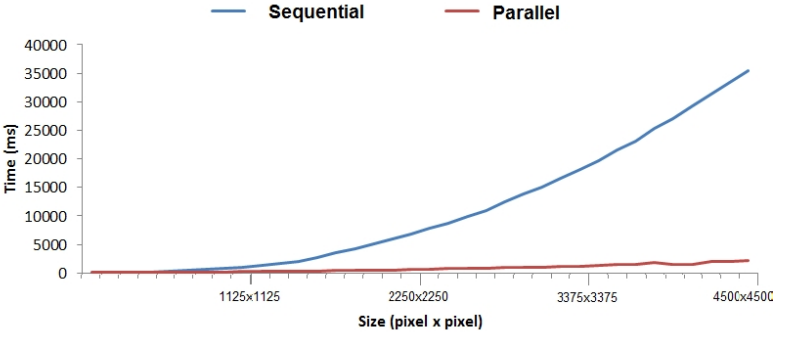

Fig. 8. A comparison of our implementation vs. the sequential one by taking 90 steps in the Guo & Hall algorithm

4 Conclusions

Computer vision is a hard task and a challenge in the next years. Classical sequential algorithms need to be revisited and adapted to the novel technologies, but the new developments also need the support of deep theoretical foundations. Using Cellular Automata is not a new concept in the study of digital images (see, for example, [18]) but the intrinsic parallelism of CA could not be effectively explored due to the limitations of sequential computers. Only recently, the new architectures of GPU has started to be studied as a tool for realistic implementations (see, e.g., [12]). This paper goes in this line by proposing a new implementation of one of the basic pre-processing problems for image analysis, the Guo & Hall algorithm. The quick development of new hardware architectures will provide in the next years a challenge for the effective parallel implementation of many other approaches in Image Analysis.

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. AB acknowledges the support of the project MTM2009-12716 of the Ministerio de Educación y Ciencia, the project EHU09/04 and the "Computational Topology and Applied Mathematics" PAICYT research group FQM-296.

References

1. Arcelli, C., di Baja, G.S.: Euclidean skeleton via centre-of-maximal-disc extraction. Image and Vision Computing 11(3), 163–173 (1993)
2. Attali, D., Boissonnat, J.D., Edelsbrunner, H.: Stability and Computation of Medial Axes - a State-of-the-Art Report. In: Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration, ch. 6, pp. 109–125. Springer, Heidelberg (2009)
3. di Baja, G.S., Thiel, E.: Skeletonization algorithm running on path-based distance maps. Image and Vision Computing 14(1), 47–57 (1996)
4. Biasotti, S., Attali, D., Boissonnat, J.-D., Edelsbrunner, H., Elber, G., Mortara, M., Baja, G.S., Spagnuolo, M., Tanase, M., Veltkamp, R.: Skeletal structures. In: Floriani, L., Spagnuolo, M. (eds.) Shape Analysis and Structuring. Mathematics and Visualization, pp. 145–183. Springer, Heidelberg (2008)
5. Blum, H.: An associative machine for dealing with the visual field and some of its biological implications. Computer and Mathematical Sciences Laboratory, Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force (1962)
6. Blum, H.: An associative machine for dealing with the visual field and some of its biological implications. In: Bernard, E.E., Kare, M.R. (eds.) Biological Prototypes and Synthetic Systems, vol. 1, pp. 244–260. Plenum Press, New York (1962); Proceedings of the 2nd Annual Bionics Symposium, held at Cornell University (1961)
7. Bräunl, T.: Parallel image processing. Springer (2001)
8. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Communications of the ACM 32, 359–373 (1989)

9. Hernandez, G., Herrmann, H.J.: Cellular-automata for elementary image-enhancement. Graphical Models and Image Processing 58(1), 82–89 (1996)
10. Kari, J.: Theory of cellular automata: A survey. Theoretical Computer Science 334(1-3), 3–33 (2005)
11. Kari, L., Rozenberg, G.: The many facets of natural computing. Communications of the ACM 51(10), 72–83 (2008)
12. Kauffmann, C., Piché, N.: A cellular automaton framework for image processing on GPU. In: Yin, P.Y. (ed.) Pattern Recoginition, pp. 353–375. InTech (2009)
13. Klette, R., Ahn, J., Haeusler, R., Herman, S., Huang, J., Khan, W., Manoharan, S., Morales, S., Morris, J., Nicolescu, R., Ren, F., Schauwecker, K., Yang, X.: Advance in vision-based driver assistance. In: 2011 International Conference on Electric Technology and Civil Engineering (ICETCE), pp. 987–990 (April 2011)
14. Mohapatra, A.G.: Computer vision based smart lane departure warning system for vehicle dynamics control. Sensors & Transducers Journal 132(9), 122–135 (2011)
15. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A., Purcell, T.J.: A survey of general-purpose computation on graphics hardware. Computer Graphics Forum 26(1), 80–113 (2007)
16. Saeed, K., Tabedzki, M., Rybnik, M., Adamski, M.: K3M: A universal algorithm for image skeletonization and a review of thinning techniques. Applied Mathematics and Computer Science 20(2), 317–335 (2010)
17. de Saint Pierre, T., Milgram, M.: New and efficient cellular algorithms for image processing. CVGIP: Image Understanding 55(3), 261–274 (1992)
18. Selvapeter, P.J., Hordijk, W.: Cellular automata for image noise filtering. In: NaBIC, pp. 193–197. IEEE (2009)
19. Siddiqi, K., Pizer, S.M.: Medial representations: mathematics, algorithms and applications. In: Computational Imaging and Vision. Springer (2008)
20. Wolfram, S.: Cellular Automata and Complexity: Collected Papers. Perseus Books Group (1994)
21. NVIDIA Corporation. NVIDIA CUDA^{tm} Programming Guide, `http://www.nvidia.com/object/cuda_home_new.html`