Chapter 3
Skeletonizing Digital Images with Cellular Automata

Daniel Díaz-Pernil, Francisco Peña-Cantillana, and Miguel A. Gutiérrez-Naranjo

Abstract. The skeletonization of an image consists of converting the initial image into a more compact representation. In general, the skeleton preserves the basic structure and, in some sense, keeps the *meaning*. The most important features concerning a shape are its *topology* (represented by connected components, holes, etc.) and its *geometry* (elongated parts, ramifications, etc.), thus they must be preserved. Skeletonization is usually considered as a pre-processing step in pattern recognition algorithms, but its study is also interesting by itself for the analysis of line-based images such as texts, line drawings, human fingerprints classification or cartography.

Since the introduction of the concept by Blum in 1962 under the name of medial axis transform, many algorithms have been published in this topic 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. In this chapter, we focus on how the skeletonization of an image can be studied in the Cellular Automata framework and, as a case study, we consider in detail the Guo and Hall skeletonizing algorithm.

3.1 Introduction

Skeletonization is one of the approaches for representing a shape with a small amount of information by converting an image into a more compact representation and keeping the meaningful features. The conversion should remove redundant

Daniel Díaz-Pernil
Research Group on Computational Topology and Applied Mathematics,
Department of Applied Mathematics, University of Seville, Spain
e-mail: sbdani@us.es

Francisco Peña-Cantillana · Miguel A. Gutiérrez-Naranjo
Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence, University of Seville, Spain
e-mail: frapencan@gmail.com, magutier@us.es

P. Rosin et al. (eds.), *Cellular Automata in Image Processing and Geometry*,
Emergence, Complexity and Computation 10,
DOI: 10.1007/978-3-319-06431-4_3, © Springer International Publishing Switzerland 2014

information, but it should also keep the basic structure. The concept of skeleton was introduced by Blum in [10, 11], under the name of medial axis transformation, but other previous approaches can be found in the literature (see, e.g., [16, 28]). The skeleton of an image is useful to characterize objects by compact representation while preserving the connectivity and topological properties of any image. The most important features concerning a shape are its *topology* (represented by connected components, holes, etc.) and its *geometry* (elongated parts, ramifications, etc.), thus they must be preserved.

Currently, there are many different definitions of the skeleton of a black and white image and, according to Saeed *et al.* [39], more than one thousand algorithms have been published on image skeletonization. Nevertheless, roughly speaking, we can say that the image B is a skeleton of the black and white image A, if the former has fewer black pixels than the latter, preserves its topological properties and, in some sense, keeps its *meaning*. Figure 3.1 illustrates this idea. The skeletonized image keeps the *meaning* of the original one and it uses fewer black pixels. It keeps the basic geometry of the original image and also its topology. Let us remark that the white regions inside the hand-made words are also white regions in the skeletonized one and the connectedness is preserved.

As pointed out by Rosenfeld [36], the concept of skeletonizing[1] is not easy to be defined mathematically; however it seems reasonable to require that any skeletonizing algorithm should preserve the connectedness for both objects and their complement; leave unchanged 1-pixel wide and isolated points; and change objects whose length and width are both greater than 1.

From the computational side, a digital image can be roughly defined as a function from a two dimensional surface which maps each point from the surface onto a set of attributes as brightness or color. Technically, a digital image can be considered as a bi-dimensional array of $n \times m$ pixels. Each pixel can be characterized by a triplet (i, j, a) (usually written as a_{ij}) where (i, j) represents its position in the array and a encodes brightness, color or any other feature associated to the position (i, j). As in many other image processing algorithms, the basic procedure for skeletonizing an image involve a discrete transformation of the features associated with the position (i, j) according to the current values of such features (i.e., the current value of a) together with the values of the features of the neighbour positions. From this point of view, a cellular automaton can be considered as a natural computer device for such transformations [23, 40]. As usual, pixels are identified with cells of the cellular automaton and the encoding a represent the current *state* of the pixel. In many cases, the transformation of all the pixels can be done in parallel, since the *state* of a pixel at the step i only depends on the *states* of a set of pixels at the step $i - 1$.

Such parallelism in skeletonizing algorithms has been broadly studied (see, e.g., [21, 32, 43, 45]). The development of new hardware architectures has also contributed to new parallel implementations of these algorithms [19, 24, 26]. Recently, a parallel implementation of a cellular automata skeletonizing algorithm developed by

[1] Rosenfeld used the word *thinning* instead of *skeletonizing*. In this paper, both terms are considered synonymous. Nevertheless, in the literature many different definitions of *skeletonizing* and *thinning* can be found, not all of them equivalent.

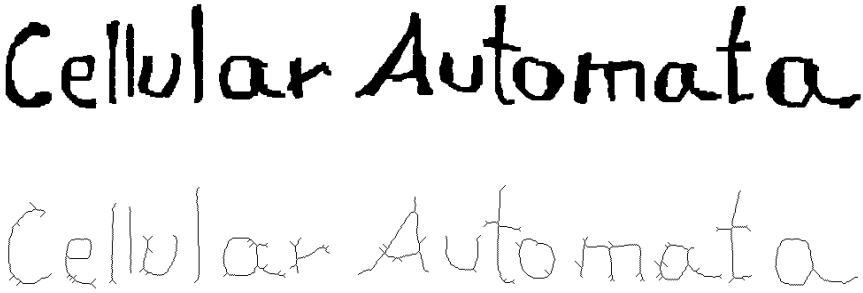


Fig. 3.1 Hand-written words and their skeletonization. This skeletonization has been performed with the parallel implementation on GPU of the cellular automata adaptation of the Guo and Hall algorithm presented in [35].

using a device architecture called Compute Unified Device Architecture (CUDA™) has been presented [35]. CUDA™ 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). In Section 3.5, we explore how this new computer architecture can be used in order to improve the efficiency of the algorithms from a practical point of view.

Skeletonization has been found useful for data compression and pattern recognition in a wide range of applications in the industrial and scientific fields. It is usually considered as a pre-processing step in pattern recognition algorithms, but its study is also interesting by itself for the analysis of line-based images such as coronary arteries [17], human fingerprints classification [27], cartography [29], data compression and data storage [20], automated inspection of printed circuit boards [44] or optical character recognition (OCR) [42] among others.

The chapter is organized as follows: Firstly, a brief overview on different skeletonizing algorithms is presented (Section 3.2). In Section 3.3, as an example, the Guo and Hall skeletonizing algorithm is studied in detail. Next, some hints on a general skeletonizing algorithm on cellular automata are shown (Section 3.4) and, as a case study, the Guo and Hall algorithm is adapted to cellular automata. In Section 3.5, some hints for a parallel implementation in CUDA are presented. Finally, some examples and some conclusions are provided.

3.2 Skeletonizing Algorithms

The first definition of the skeleton of a region was provided by Blum as the medial axis transformation (MAT) [10, 11]. According to the original definition, in order to find the MAT region with border B from a region R, the closest neighbor in B

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

of each pixel in R should be found. If the pixel has more than one such neighbor, it is said to belong to the medial axis (skeleton) of R. The definition of *closest* depends, of course, on the definition of distance. Obviously, the implementation of such definition is typically prohibitive computationally. Many algorithms have been proposed for improving computational efficiency while at the same time attempting to produce a medial axis representation. Such skeletonizing algorithms iteratively delete edge points of a region subject to the constraints that deletion of these points (1) does not remove end points, (2) does not break connectedness and (3) does not cause excessive erosion of the region.

Due to the enormous amount of algorithms, it is not easy to classify them. Different criteria can be used in a a preliminary approach. A first classification can be done according to the method used to obtain the skeleton. Following this criterion, the algorithms are split into iterative and non-iterative. A second classification method focuses on the properties which a black pixel must satisfy in order to be marked as erasable. In this way, the properties can be *local* or *global*.

According to the first criterion, the skeletonizing algorithms can be classified as either iterative or non-iterative. In iterative methods, skeletonizing algorithms produce a skeleton by examining and deleting contour pixels through an iterative process in either sequential or parallel way. Parallel algorithms may also be further classified according to their performance, i.e., in 4-, 2-, or 1-subcycle manners. The latter (1-subcycle parallel algorithms) have always received more considerable attention in the research area of parallel skeletonizing as they have reduced the computation time in the number of iterations, and that is why they are sometimes called one-pass or fully parallel algorithms [14, 15, 22]. In sequential skeletonizing algorithms, contour points are examined for deletion in a predetermined order. In parallel skeletonizing algorithms, pixels are examined for deletion on the basis of results obtained only from the previous iteration. That is why parallel skeletonizing algorithms are suitable for implementation in parallel processors. Non-iterative (non-pixel based) skeletonizing algorithms produce a certain median or centre line of the pattern to be thinned directly in one pass, without examining all the individual pixels.

According to the second criterion, (see [19]), two different methods of pattern analysis can be applied to determine the skeleton of an image or scene: Global Pattern Analysis, where pixels are labeled depending on their distance from the contours and Local Pattern Analysis, based on the repetition of the simultaneous deletion of border points verifying certain conditions. This classification is not strict, and some *hybrid* methods can be found in the literature, as Liu's method [30, 31], based on the notion of *cell complex*. This method combines an iterative process where *outer* cells are removed with two difference measures which provide some ideas of the size of the maximum disk inscribed in the object.

Since Dinneenn [16] found in the 1950s that an averaging operation over a square window with a high threshold resulted in a skeletonizing of the input image and later Blum presented his definition of medial axis transformation [10, 11], many different authors have contributed to the skeletonizing theory. After some attempts of defining the skeleton of an image and different skeletonizing algorithms, (e.g. [25, 38]),

Rosenfeld [36] was the first to evaluate the necessary and sufficient conditions for preserving topology while deleting border points in parallel process. Only a few years later, Pavlidis [34] proposed a combination of parallel and sequential operations.

In 1980s, many other algorithms were proposed (see, e.g., [3, 18, 45]). Among them, it is Baruch's remarkable algorithm [7]. It is a non iterative, non pixel-based algorithm. The skeleton is produced in one pass, by line following. Another important contribution is the Guo and Hall algorithm, which will be showed in detail in Section 3.3. In this algorithm, the contour pixels are examined for deletion in an iterative process and it has been the basis for further refinements (see, e.g., [46]). There are many different approaches to the problem of skeletonizing and a detailed survey of the different methods is out of the scope of this chapter[3]. Among the different research areas around the skeletonization problem, we can cite the studies based on distance transform of the shape and skeleton pruning based on branch analysis (see, for example, [4–6, 9, 41]). The research of skeletonizing algorithms also includes different computational models, as the algorithm presented by Altuwaijri and Bayoumi [2] where self-organizing neural networks are used or the use of skeletonizing in color images [33].

As pointed out above, there exists an enormous amount of skeletonizing algorithms. Generally, each of the presented algorithms has its own advantages and disadvantages, and each has its applications where it performs better than others. Therefore, it is often difficult to directly compare the results.

3.3 Guo and Hall Algorithm

As an example of skeletonizing algorithm, we will see the implementation of a classical algorithm, the Guo and Hall algorithm [21, 22]. It is a so-called 1-subcycle parallel algorithm or fully parallel algorithm. It is an iterative edge-point erosion algorithm where a 3×3 window is considered around each pixel of the image with a set of rules applied to the contents of the window. In a sequential simulation of the algorithm, a *unique* window is moved along the image, whereas in the parallel one, *all the windows* are considered simultaneously. The skeleton is obtained by an iterative procedure of skeletonizing: the border points are removed as long as they are not considered significant. The remaining set of points is called the *skeleton*.

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. One of the sections is composed by the pixels a_{ij} such that $i + j$ is even. Alternatively, the second subsection corresponds to the pixels a_{ij} such that $i + j$ is odd. The algorithm consists on 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. This is repeated until there are no redundant pixels left.

[3] A good introduction can be found in [39].

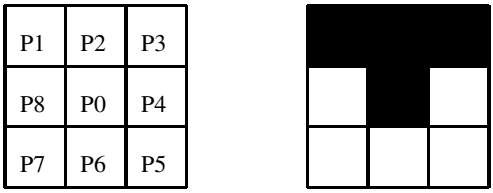


Fig. 3.2 (a) Enumeration of the pixels in a 3×3 neighborhood. (b) An example of 3×3 neighborhood where the central pixel $P0$ has $B(P0) = 3$ and $C(P0) = 1$.

Given a pixel $P0$, a clockwise enumeration $P1, \ldots, P8$ of its eight neighbor pixels is considered, as shown in Fig. 3.2 (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 subcycle, two parameters are evaluated:

$$\begin{aligned} B(P0) &= \textstyle\sum_{i=1}^{i=8} Pi \\ C(P0) &= (\neg P2 \wedge (P3 \vee P4)) + (\neg P4 \wedge (P5 \vee P6)) \\ &\quad + (\neg P6 \wedge (P7 \vee P8)) + (\neg P8 \wedge (P1 \vee P2)) \end{aligned}$$

$B(P0)$ 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. If the pixel is surrounded by eight black pixels, the connectivity is also 0. 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. 3.2 (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 subcycle iteration.

3.4 Cellular Automata

Cellular automata (CA) are dynamical systems discrete in time and space. 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 [13, 23, 37, 40].

The dynamics of CA has been extensively studied across a variety of disciplines from different perspectives (see, e.g., [1, 8, 12]). 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. 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, we will consider as *cellular space* a rectangular grid $(n+2) \times (m+2)$ with 8-adjacency. We will consider a cell of the cellular space for each pixel on the image plus two extra columns and rows surrounding such cells. The intuition behind these extra pixel lines is to consider that they are *white pixels*, which do no affect to the skeletonizing process. The states in these extra cells never change and we will consider that the rules are not applied on them, but they contribute to the neighborhood of the border pixels of the original image.

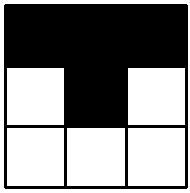
With respect to the set of states, they must encode the information of each pixel along all the steps of the algorithm. The basic information for the skeletonizing process of a black and white image is, of course, the color of the pixel, but, depending on the algorithm, different features can be associated to the pixel. If these features change dynamically along the skeletonizing process, they should be encoded in the set of states.

In order to fix ideas, we will consider the Guo and Hall algorithm. On the one hand, the set of states should inform about the *color* B or W (black or white) of the pixel at each discrete step, but, according to the algorithm, the set of pixels are split into two subsections and only one of them is evaluated at each step. In this way, the state of a cell at time t should inform if the cell corresponds to an evaluable pixel or to a non-evaluable pixel. Putting together both pieces of information, the color and the evaluable situation (E for evaluable and N for non evaluable), we obtain four possible states for each cell of the cellular space: BE, WE, BN and WN.

Bearing in mind this interpretation for the states, the initial state of each cell is determined by the color of the corresponding pixel in the input image and the corresponding sub-section. In the initial configuration, we will take the pixels of the first sub-section, corresponding to the pixels a_{ij} such that $i+j$ is even, are *evaluable* and the pixels corresponding to the second sub-section, i.e., pixels a_{ij} such that $i+j$ is odd, are non evaluable[4] (see Fig. 3.3).

In the description of the set of rules, it is necessary to describe the conditions necessary for changing the state of a cell. Such conditions are taken directly from

[4] If the second subsection is taken as evaluable in the initial configuration, the pixels of the final skeleton are different.



(a)

WN	WE	WN	WE	WN
WE	BN	BE	BN	WE
WN	WE	BN	WE	WN
WE	WN	WE	WN	WE
WN	WE	WN	WE	WN

(b)

Fig. 3.3 A toy example of black and white image and the initial configuration of the associated CA for skeletonizing it according to the Guo and Hall algorithm

the Guo and Hall algorithm. A cell will change its state if the corresponding pixel in the image satisfies the Guo and Hall conditions. Since each cell can be in one of four different states, a full description of the rules needs to consider all the possibilities of the 3×3 neighborhood, i.e., $4^9 = 2^{18}$ rules. Obviously, they are too much from a practical point of view. Nevertheless, we can provide an intensive description.

- If the central pixel is in the state WN, the next state is WE, regardless the states of the surrounding pixels.
- If the central pixel is in the state WE, the next state is WN, regardless the states of the surrounding pixels.

The intuition is clear. If the cell corresponds to a white pixel, it remains white to the end of the process. The unique change is that the pixel alternates the condition of evaluable and non evaluable in each step.

- If the central pixel is in the state BN, the next state is BE, regardless the remaining cells of the neighborhood.

If a cell correspond to a black pixel and the pixel belongs to the non evaluable section, it remains black, but in the next step it will be evaluable.

With these descriptions, it only remains to describe the rules in case of the central pixel is BE. In this case, it should be evaluated according to the Guo and Hall conditions in order to decide if the next state will be BN or WN. Nevertheless, the number of possibilities is enormous. By fixing the state of the central pixel, four possibilities for the remaining eight cells, i.e., $4^8 = 2^{16}$ possibilities, should be considered.

In order to reduce the number of different CA rules, we observe than the decision of change the current state BE (to the state BN or WN) only depends on the color of the surrounding pixels and not on the evaluable or non evaluable condition. This consideration reduces drastically the number of possibilities, since only $2^8 = 256$ cases should be considered.

In order to encode each of these cases, we can use the enumeration of the pixels used in the Section 3.3 to represent the neighborhood of the pixel $P0$ in (i, j). Given

$$DEL = \left\{ \begin{array}{cccccccccccc} 6 & 12 & 14 & 20 & 22 & 24 & 28 & 30 & 48 & 52 & 54 & 56 \\ 60 & 62 & 80 & 84 & 86 & 88 & 92 & 94 & 96 & 112 & 116 & 118 \\ 120 & 124 & 126 & 192 & 208 & 212 & 214 & 216 & 220 & 222 & 224 & 240 \\ 244 & 246 & 248 & 252 & 258 & 260 & 262 & 268 & 270 & 276 & 278 & 284 \\ 286 & 308 & 310 & 316 & 318 & 320 & 322 & 324 & 326 & 332 & 334 & 336 \\ 338 & 344 & 346 & 352 & 354 & 356 & 358 & 364 & 366 & 368 & 370 & 376 \\ 378 & 384 & 386 & 388 & 390 & 396 & 398 & 404 & 406 & 412 & 414 & 436 \\ 438 & 444 & 448 & 450 & 452 & 454 & 460 & 462 & 464 & 466 & 472 & 474 \\ 480 & 482 & 484 & 486 & 492 & 496 & 498 & 504 & & & & \end{array} \right\}$$

Fig. 3.4 Set of the codings for erasing a black pixel in an evaluable section

a cell (i, j) at a time t, the states of its eight surrounding pixels will be represented as a list $[H1, \ldots, H8]$, where, for $r \in \{1, \ldots, 8\}$, $Hr = 1$ if the state of the cell is BE or BN (i.e., if the cell corresponds to a black pixel) and $Hr = 0$ if the state of the cell is WE or WN (a white pixel). This representation of the neighborhood can be done in a more compact way, by encoding the neighborhood as a number[5] in $\{2i : i \in \{0, \ldots, 255\}\}$:

$$cod(i, j) = \sum_{r=1}^{8} Hr \times 2^r$$

For example, in Fig. 3.2 (b), the states of the cells surrounding the central one can be encoded as $[1, 1, 1, 0, 0, 0, 0, 0,]$, or, shortly, $2^1 + 2^2 + 2^3 = 14$.

Since the decision of removing a black pixel (changing to white) depends on its surrounding pixels and there is a bijective correspondence among the sets of all the possibilities and the possible encodings $\{2i : i \in \{0, \ldots, 255\}\}$. It is easy to check that the cell in (i, j) must change its state to WN (i.e., the corresponding black pixel must be deleted) if the encoding of the surrounding cells belong to the set DEL showed in Fig. 3.4. A quick check on the set DEL will decide if the state BE changes to WN or BN.

3.5 Parallel Implementation

The parallel iterative algorithms are the result of processing a pixel at the i-th iteration just depending on the value of the neighbor pixels in the $(i-1)$-th iteration. Thus parallel iterative algorithms can process all pixels in the image simultaneously.

As pointed above, the parallelism in skeletonizing algorithms has been deeply studied, but the real implementation is associated to the development of new parallel hardware [19, 24, 26]. Recently, in [35], a parallel implementation of a cellular automata skeletonizing algorithm developed by using CUDA™ has been presented.

[5] Similar ideas are also used in [39].

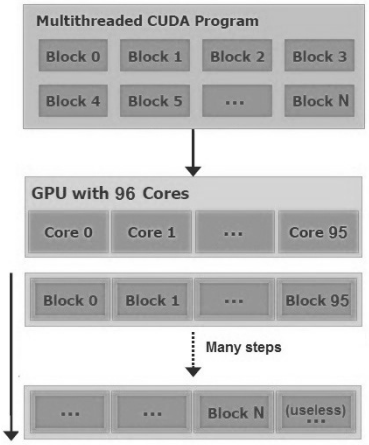


Fig. 3.5 Scheme of threads for the CUDA implementation

It shows some important features that make this new hardware architecture suitable for further implementations of CA algorithms.

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 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.

3.5.1 Examples

In [35], a parallel implementation of the Guo and Hall algorithm for CA based on the principles described above was presented. It has been implemented by using Microsoft Visual Studio 2008 Professional Edition (C++) with the plug-in 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 to use the *nvcc compiler* of CUDA™ Toolkit and some libraries from openCV for image input and output.

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

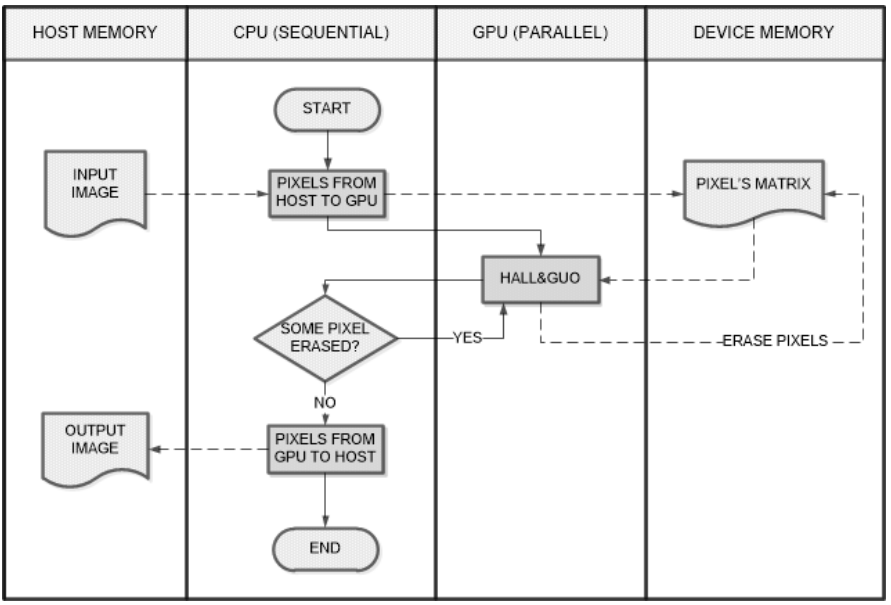


Fig. 3.6 Flowchart of the implementation on CUDA™ of the CA algorithm

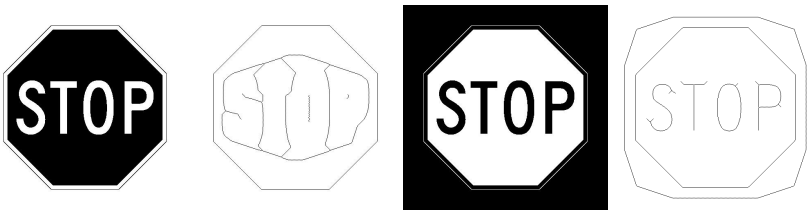


Fig. 3.7 The binarized STOP signal, its skeletonization, its inverted binarization and its skeletonization

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. Figure 3.6 shows a flowchart of the implementation on CUDA of the algorithm.

Next, we show the results of some experiments of skeletonizing with our parallel CA implementation of the Guo and Hall algorithm. As a first example[6], Fig. 3.7, shows the skeletonization of the STOP traffic signal. This example illustrates an old problem in black and white images. In such images, the *meaning* is usually associated to black pixels, whereas white pixels constitute the background. In this

[6] These examples are borrowed from [35].

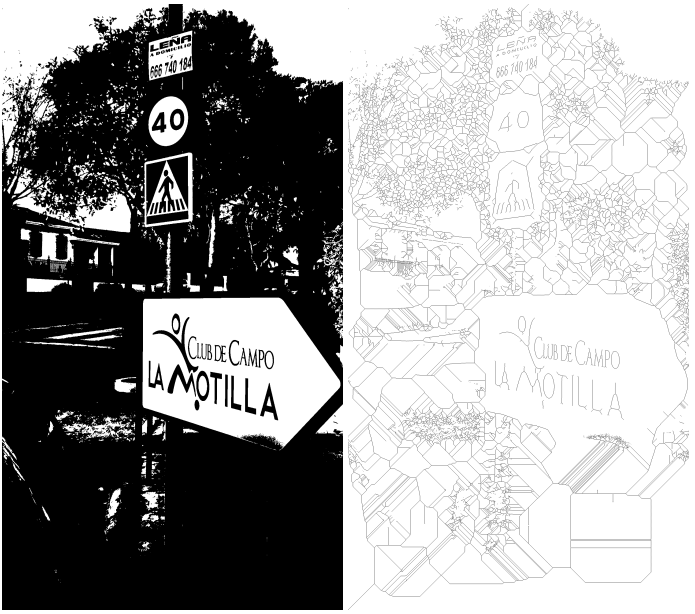


Fig. 3.8 The binarization of a real photograph and its skeleton

case, the skeletonization of the image with inverted colors is more *meaningful*. In Fig. 3.8, we can see two black and white images of 789×1317 pixels. On the left, the result of the binarization of a real photograph and its skeletonization on the right.

A last example is presented in Fig. 3.9. On the left, a fingerprint is shown and on the right, its skeletonization. The basic problem with fingerprints is to determine whether two fingerprints come from the same finger. There exist multiple algorithms that do fingerprint matching in many different ways. Some methods involve matching minutiae points between the two images, while others look for similarities in the

Fig. 3.9 A fingerprint and its skeleton

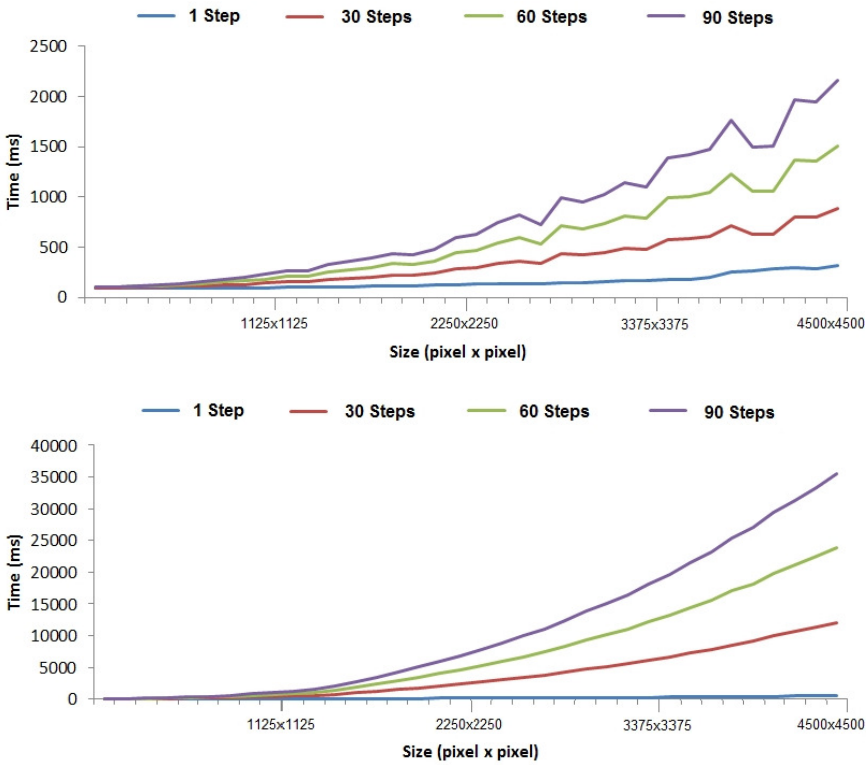


Fig. 3.10 Experimental time obtained for the Guo and Hall algorithm applied on 36 totally black images of $n \times n$ pixels, from $n = 125$ to $n = 4500$ with a regular increment of 125 pixels. Top image shows the time of our parallel implementation in CA. Bottom image shows the time for a sequential implementation.

bigger structure of the fingerprint. In many cases, the thickness of each line print is not important and the skeletonized image provides the same information as the original one.

We finish this section by showing the results of some experiments performed with our CA implementation. We have taken 36 totally black images of $n \times n$ pixels, from $n = 125$ to $n = 4500$ with a regular increment of 125 pixels of side. Figure 3.10 (top) shows the time in milliseconds of the application of our implementation of the Guo and Hall algorithm in CA for 1, 30, 60 and 90 steps in the skeletonizing process. Figure 3.10 (bottom) shows the same study for a sequential implementation of the algorithm. Finally, Figure 3.11 shows a comparison of our implementation vs. the sequential one by taking 90 steps in the Guo and Hall algorithm.

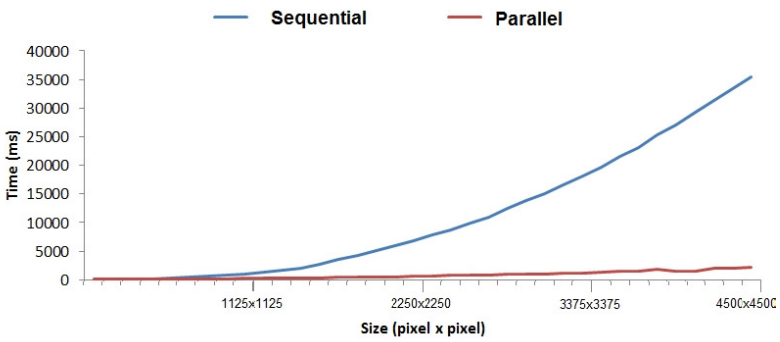


Fig. 3.11 A comparison of our implementation vs. the sequential one by taking 90 steps in the Guo and Hall algorithm

3.6 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. On the one hand, it is necessary to develop new algorithms in the framework of CA or other computational paradigms which can put a new light on classical problems. A deep study of the properties of such algorithms can improve their practical efficiency and accelerate their use in digital image industry.

On the other hand, 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. In this way, the theoretical study should be also oriented to the most recent advances in hardware architecture. In this chapter, we have considered the implementation on GPUs. It is a good alternative for future implementations for CA algorithms, but many other possibilities should be explored. A strong link between the research on theoretical models and the development of new hardware architectures is the key for realistic answers to the challenges of the future in digital image areas.

Acknowledgements. MAGN acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

References

1. Acerbi, L., Dennunzio, A., Formenti, E.: Conservation of some dynamical properties for operations on cellular automata. Theoretical Computer Science 410(38-40), 3685–3693 (2009)

2. Altuwaijri, M., Bayoumi, M.: A new thinning algorithm for Arabic characters using self-organizing neural network. In: 1995 IEEE International Symposium on Circuits and Systems, ISCAS 1995, vol. 3, pp. 1824–1827 (1995)
3. Ammann, C., Sartori Angus, A.: Fast thinning algorithm for binary images. Image Vision and Computing 3(2), 71–79 (1985)
4. Arcelli, C., di Baja, G.S.: Euclidean skeleton via centre-of-maximal-disc extraction. Image and Vision Computing 11(3), 163–173 (1993)
5. 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)
6. di Baja, G.S., Thiel, E.: Skeletonization algorithm running on path-based distance maps. Image and Vision Computing 14(1), 47–57 (1996)
7. Baruch, O.: Line thinning by line following. Pattern Recognition Letters 8(4), 271–276 (1988)
8. Betel, H., Flocchini, P.: On the relationship between fuzzy and boolean cellular automata. Theoretical Computer Science 412(8-10), 703–713 (2011)
9. 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)
10. 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)
11. Blum, H.: An associative machine for dealing with the visual field and some of its biological implications. In: Computer and Mathematical Sciences Laboratory, Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research. United States Air Force (1962)
12. Cattaneo, G., Dennunzio, A., Margara, L.: Solution of some conjectures about topological properties of linear cellular automata. Theoretical Computer Science 325(2), 249–271 (2004), Theoretical Aspects of Cellular Automata
13. Chauhan, S.: Survey paper on training of cellular automata for image. International Journal of Engineering and Computer Science 2(4), 980–985 (2013)
14. Chen, Y.S.: The use of hidden deletable pixel detection to obtain bias-reduced skeletons in parallel thinning. In: Proceedings of the 13th International Conference on Pattern Recognition, vol. 2, pp. 91–95. IEEE Computer Society, Washington, DC (1996)
15. Chen, Y.S., Hsu, W.H.: Systematic approach for designing 2-subcycle and pseudo 1-subcycle parallel thinning algorithms. Pattern Recognition 22(3), 267–282 (1989)
16. Dinneen, G.P.: Programming pattern recognition. In: Proceedings of the Western Joint Computer Conference, AFIPS 1955 (Western), pp. 94–100. ACM, New York (1955)
17. Dufresne, T.E., Sarwal, A., Dhawan, A.P.: A gray-level thinning method for delineation and representation of arteries. Computerized Medical Imaging and Graphics 18(5), 343–355 (1994)
18. Favre, A., Keller, H.: Parallel syntactic thinning by recoding of binary pictures. Computer Vision, Graphics, and Image Processing 23(1), 99–112 (1983)
19. Gil Montoya, M., Garcia, I.: Implementation of parallel thinning algorithms on multicomputers: analysis of the work load balance. In: Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing, PDP 1998, pp. 257–263 (1998)
20. González, R.C., Woods, R.E.: Digital image processing. Pearson/Prentice Hall (2008)

21. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Communications of the ACM 32, 359–373 (1989)
22. Guo, Z., Hall, R.W.: Fast fully parallel thinning algorithms. CVGIP: Image Understanding 55, 317–328 (1992)
23. Hernandez, G., Herrmann, H.: Cellular-automata for elementary image-enhancement. Graphical Models and Image Processing 58(1), 82–89 (1996)
24. Heydorn, S., Weidner, P.: Optimization and performance analysis of thinning algorithms on parallel computers. Parallel Computing 17(1), 17–27 (1991)
25. Hilditch, C.: An application of graph theory in pattern recognition. Machine Intelligence 3, 325–347 (1968)
26. Holt, C., Stewart, A.: A parallel thinning algorithm with fine grain subtasking. Parallel Computing 10(3), 329–334 (1989)
27. Hongbin, P., Junali, C., Yashe, Z.: Fingerprint thinning algorithm based on mathematical morphology. In: 8th International Conference on Electronic Measurement and Instruments, ICEMI 2007, pp. 2–618–2–621 (2007)
28. Kirsch, R.A., Cahn, L., Ray, C., Urban, G.H.: Experiments in processing pictorial information with a digital computer. In: Papers and Discussions Presented at the December 9-13, Eastern Joint Computer Conference: Computers with Deadlines to Meet, IRE-ACM-AIEE 1957 (Eastern), pp. 221–229. ACM, New York (1958)
29. Lee, K.H., Cho, S.B., Choy, Y.C.: Automated vectorization of cartographic maps by a knowledge-based system. Engineering Applications of Artificial Intelligence 13(2), 165–178 (2000)
30. Liu, L.: 3D thinning on cell complexes for computing curve and surface skeletons. Washington University (2009)
31. Liu, L., Chambers, E.W., Letscher, D., Ju, T.: A simple and robust thinning algorithm on cell complexes. Computer Graphics Forum 29(7), 2253–2260 (2010)
32. Lü, H.E., Wang, P.S.P.: A comment on a fast parallel algorithm for thinning digital patterns. Communications of the ACM 29(3), 239–242 (1986)
33. Nedzved, A., Ilyich, Y., Ablameyko, S., Kamata, S.: Color thinning with applications to biomedical images. In: Skarbek, W. (ed.) CAIP 2001. LNCS, vol. 2124, pp. 256–263. Springer, Heidelberg (2001)
34. Pavlidis, T.: Algorithms for Graphics and Image Processing. Digital system design series. Computer Science Press (1982)
35. Peña-Cantillana, F., Berciano, A., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A.: Parallel skeletonizing of digital images by using cellular automata. In: Ferri, M., Frosini, P., Landi, C., Cerri, A., Di Fabio, B. (eds.) CTIC 2012. LNCS, vol. 7309, pp. 39–48. Springer, Heidelberg (2012)
36. Rosenfeld, A.: A characterization of parallel thinning algorithms. Information and Control 29(3), 286–291 (1975)
37. Rosin, P.L.: Training cellular automata for image processing. IEEE Transactions on Image Processing 15(7), 2076–2087 (2006)
38. Rutovitz, D.: Pattern recognition. Journal of the Royal Statistical Society 129, 504–530 (1966)
39. 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)
40. de Saint Pierre, T., Milgram, M.: New and efficient cellular algorithms for image processing. CVGIP: Image Understanding 55(3), 261–274 (1992)
41. Siddiqi, K., Pizer, S.: Medial representations: mathematics, algorithms and applications. Computational imaging and vision. Springer (2008)

42. Smith, S.J., Bourgoin, M.O., Sims, K., Voorhees, H.L.: Handwritten character classification using nearest neighbor in large databases. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(9), 915–919 (1994)
43. Suzuki, S., Abe, K.: Binary picture thinning by an iterative parallel two-subcycle operation. Pattern Recognition 20(3), 297–307 (1987)
44. Ye, Q.Z., Danielsson, P.E.: Inspection of printed circuit boards by connectivity preserving shrinking. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(5), 737–742 (1988)
45. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Communications of the ACM 27(3), 236–239 (1984)
46. Zhang, Y.Y., Wang, P.S.P.: A parallel thinning algorithm with two-subiteration that generates one-pixel-wide skeletons. In: Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, pp. 457–461 (1996)