[Proc. SPIE/SPSE Vol. 1451, "Nonlinear Image Processing II", pp. 158–170, San Jose CA, February 1991]

New trends in morphological algorithms

Luc Vincent

Harvard University, Division of Applied Sciences
Pierce Hall, Cambridge MA 02138, USA

Abstract

This paper describes some of the most recent algorithmic techniques of mathematical morphology. The classical parallel and sequential methods mostly involve numerous scannings of *all* the image pixels and are thus inefficient on conventional computers. To get rid of this drawback, the key-idea is to consider, at each step, only the pixels whose value may be modified. Two classes of algorithms rely on this principle: the first one realizes an encoding of the object boundaries as loops which are then propagated in the image. The second one embraces the algorithms based on breadth-first image scannings enabled by queues of pixels. The algorithms belonging to these two families are extremely efficient and particularly suited to non-specialized equipments, since they require random access to the pixels. Moreover, the "customized" image scannings they are based on allow one to develop more accurate and flexible procedures: for example, algorithms for computing exact Euclidean distance functions have been derived from the first class. On the other hand, the queue-based algorithms work in any kind of grid, in the Euclidean and geodesic cases, and extend to any dimension and even to graphs.

1 Introduction

The resolution of an image analysis problem by means of mathematical morphology techniques [17, 25] mostly involves the iteration and concatenation of numerous basic transformations [2]. They constitute the elementary bricks of the entire program and must therefore be implemented as efficiently as possible. This implementation may be approached by various methods, which shall be reviewed in this paper (see also [31]). The emphasis will be put on the most recent methods (sections 5 and 6), which prove to be extremely fast and flexible.

In the sequel, we consider binary and grayscale images I as mappings from a rectangular domain $D_I \subset \mathbb{Z}^2$ into $\mathbb{Z}$. The underlying grid $G \subset \mathbb{Z}^2 \times \mathbb{Z}^2$ provides the neighborhood relations between pixels. G is usually a square grid (in 4- or 8-connectivity) or a hexagonal one (see Fig. 1). $N_G(p)$ denotes the set of the neighbors of a pixel $p \in \mathbb{Z}^2$ according to grid G:

$$N_G(p) = \{q \in \mathbb{Z}^2 \mid (p.q) \in G\}.$$

The discrete distance associated with G is denoted d_G: $d_G(p,q)$ is the minimal length of the paths of G connecting p to q. Here, we shall mostly use the hexagonal grid because of its good behavior with respect to connectivity problems [23, page 61]. Its elementary vectors are denoted $\vec{u}_0$, $\vec{u}_1$,...,$\vec{u}_5$ and illustrated by Fig. 2. However, let us stress that except those of § 5, all the algorithms described below can be easily implemented in any kind of discrete grid.

Throughout the paper, a particular transformation called *distance function* [21, 4] is used to illustrate the different families of algorithms discussed. Recall that the distance function $dist_X$ of a set $X \subset \mathbb{Z}^2$ associates with each pixel of X its distance to the background:

$$dist_X \left(\begin{array}{ccc} X & \longrightarrow & \mathbb{Z} \\ p & \longmapsto & \min\{d_G(p,q) \mid q \notin X\} \end{array} \right) \quad (1)$$

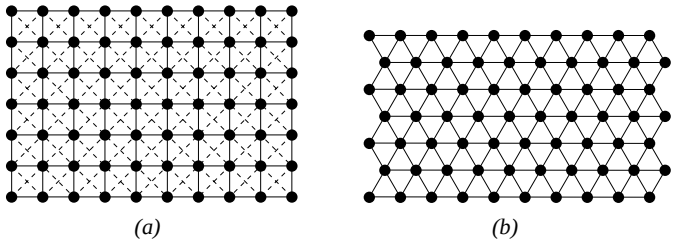

Figure 1 : *Square (a) and hexagonal (b) digital grids.*

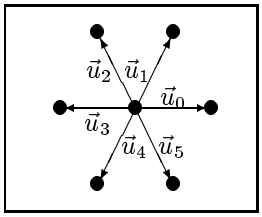

Figure 2 : *The 6 elementary vectors of the hexagonal grid.*

The distance function of a binary image I is equivalent to that of its set of *feature pixels,* i.e., pixels with value 1. In addition, we put conventionally: $\forall p \in D_I, I(p) = 0 \Rightarrow dist_I(p) = 0$. An example of distance function is shown in Fig. 3.

(a) original image *(b) distance function* *(c) level lines of this function*

Figure 3 : *Example of hexagonal distance function.*

2 What is expected from a morphological algorithm?

The **speed** of an algorithm is a crucial issue in the field of image analysis. Indeed, on the one hand, an application program is often designed to be used in routine, either on a large amount of data (e.g., in medicine), or daily (e.g., in quality control). It is then unacceptable for the execution time to be larger than a specified upper bound. On the other hand, even during the resolution of a given image analysis problem, many different possibilities have to be considered; for each of them, many transformations have to be used, often repeatedly, with parameter adjustments,

filter modifications, etc. It is therefore extremely important for the image analyst to have fast algorithms at his disposal: it considerably speeds up this development step and even allows him to explore ideas which could not be considered otherwise. For example, until recently, the use of the watershed transformation [7, 1] was impossible in practice because of its prohibitive computation time. However, the appearance of the most recent specialized architectures (e.g., the *Quantimet 570* of *Leitz*) and algorithms [31, 26] has moved it to one of the highest ranking of morphological segmentation tools.

The second important characteristic of a morphological algorithm is its **accuracy.** Most of the time, the result of an algorithm is expected to be totally exact. However, the definition of some transformations, like skeletons (see [31, chapter 7] or [32]), is sometimes not well adapted to the discrete framework. The algorithms for computing such transformations should then be designed to produce results "as close as possible" to the Euclidean one. Morphological algorithms are also expected to avoid some of the aberrations associated with the use of discrete grids, like the "cone effect" (see Fig. 4). Lastly, one often tries to compute morphological transformations in an isotropic fashion. This involves resorting to discrete distances d closer to the Euclidean one than d_G.

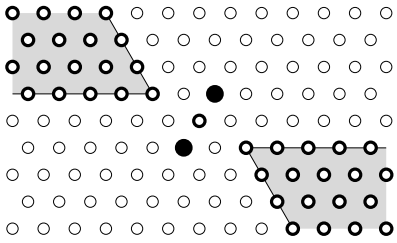

Figure 4 : *In a discrete space, the set of the pixels which are equidistant to two given connected components may be a thick area: here, according to the hexagonal distance, all the bold pixels (gray area) are equidistant to the two black ones!*

Finally, a morphological algorithm should be as **flexible** as possible. By flexibility, we hereafter refer to one of the following cases:

- the algorithm is adaptable to other grids,
- it works in both the Euclidean and geodesic [14] cases,
- it allows one to produce several transforms close to one another,
- it is adaptable to several metrics.

Such algorithms are therefore very interesting in the sense that they spare the energy of the programmer and reduce the implementation costs.

However, most algorithms cannot be extremely fast, accurate and flexible at the same time. Improving one of these characteristics is generally done to the detriment of the two remaining ones. For example, increasing the accuracy of an algorithm mostly requires additional tests and computations which affect its speed... On the other hand, morphological operations are not implemented in the same way on two different computers: on specialized architectures, such complex transformations as skeletons and watersheds will be implemented using the built-in thinning and thickening capabilities. However, such algorithms would be terribly slow on classical computers, where their execution times could approach a couple of hours!

In this paper, we are mainly concerned with conventional computers. We first consider the classical *parallel* and *sequential* algorithms. The study of their drawbacks leads us to propose new methods based on contours. We will show and illustrate in what respects these algorithms are more efficient. The methods derived from *computational geometry* [19] or involving quadtree encodings [22] shall not be considered here because of their inadaptation to morphology [31, page 22]. In the sequel, images are supposed to be represented as simple 2-d arrays. Nevertheless, some special data structures will be used to access these arrays.

3 Parallel Algorithms

This category of algorithms is the most common one in the field of morphology. A parallel algorithm typically works as follows: given an input image I, the pixels of I are scanned and the new value of the current pixel p is determined from that of the pixels in a given neighborhood $V(p)$ of p. In doing this, the following constraint is satisfied:

‖ *The new pixel values are written in an output image J different from I.*

J is then copied into I, and additional image scannings are performed until a given criterion is fulfilled, or until stability is reached.

J being different from I, the pixels can actually be scanned *in an arbitrary order.* In particular, one can imagine parallelizing the processing on some image parts, or even on all pixels, as is realized by some specialized architectures. Hereafter, a "parallel scanning" will be introduced by a sentence like:

For every pixel p of D_I, do $\{\ldots$

The parallel algorithm to determine the distance function of a binary image I in grid G is given below in a pseudo-langage. Note that in this paper, edge effects are not taken into account: the images under study are considered to be defined in the entire space $\mathbb{Z}^2$ and to take value 0 outside of their definition domain, unless otherwise mentioned.

Algorithm: parallel distance function

- – input: I, binary image,
  – output: J, grayscale image defined on D_I; $J \neq I$.
- Repeat until stability {
  For every pixel $p \in D_I$ { /⋆ actual parallel scanning ⋆/
  If $I(p) = 1$ then $J(p) \leftarrow \min\{I(q), q \in N_G(p)\} + 1$;
  Copy image J in I;
  }

This algorithm is illustrated on Figure 5. One is easily convinced that the number of scannings it requires is proportional to the largest computed distance. More generally, parallel algorithms usually require a large number of complete image scannings, sometimes several hundreds! Therefore, although these algorithms are particularly suited to some architectures, they are definitely not adapted to conventional computers.

The basic parallel algorithms (dilations, erosions, distance function, etc) are easily adapted to any grid and to n-dimensional images [9], but here again, the prohibitive execution times limit their practical interest. Moreover, the parallel computation of some more complex transformations like skeletons [5] and skeletons by influence zones (SKIZ) [13] is usually achived via iterations of parallel thinnings and thickenings. These operations involve *structuring elements* [25], i.e., pixel templates used as probes (see Golay's alphabet [10]): this is the reason why their adaptation to other grids requires cumbersome neighborhood analyses, this remark being even more true when it comes to extend these algorithms to n-dimensional data! In many cases, the very local way parallel algorithms work results in approximative results (e.g., for skeletons). However, let us mention that some of them can bring Euclidean distances into play [35].

4 Sequential Algorithms

In an attempt to reduce the number of scannings required for the computation of an image transform, sequential or recursive algorithms have been proposed [20]. They rely on the following two principles:

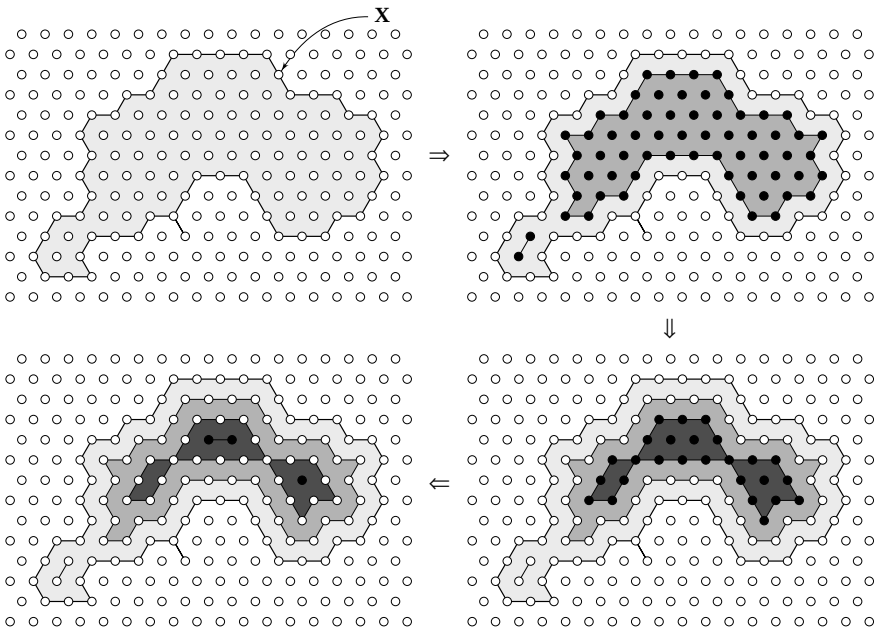

Figure 5 : *Successive steps involved in the parallel computation of a distance function.*

- *the image pixels are scanned in a predefined order, generally video or anti-video,*
- *the new value of the current pixel, determined from the values of the pixels in its neighborhood, is written directly* in the same image, *so that it is taken into account to determine the new values of the not yet considered pixels.*

Here, unlike for parallel algorithms, the scanning order is essential. This class of algorithms is very popular in the field of mathematical morphology, and a number of transformations which can be obtained sequentially are described in [15]. A sequential scanning will be introduced by:

Scan D_I in video order {
Let p be the current pixel; ...

For the hexagonal grid, the sequential distance function algorithm is the following:

Algorithm: sequential distance function

- input: I, binary image;
  /$\star$ The distance function is computed directly in I $\star$/
- Scan D_I in video order {
  Let p be the current pixel;
  If $I(p) \neq 0$ then $I(p) \leftarrow \min\{I(p+\vec{u}_1)+1, I(p+\vec{u}_2)+1, I(p+\vec{u}_3)+1\}$;
  }
- Scan D_I in anti-video order {
  Let p be the current pixel;
  If $I(p) \neq 0$ then $I(p) \leftarrow \min\{I(p), I(p+\vec{u}_4)+1, I(p+\vec{u}_5)+1, I(p+\vec{u}_0)+1\}$;
  }

In all cases, the above algorithm only requires *two* image scannings*. In comparison with the parallel one, it constitutes a clear improvement. Often, sequential algorithms are one of the best possible choices. This is true for example in the following cases: grayscale reconstruction and regional extrema [2], morphological shadowing [30], granulometry function [2]... Additionally, basic sequential algorithms such as the distance function one are easily extended to n-dimensional images [3] and can be adapted to better discrete distances [4, 6].

However, although each parallel algorithm has theorically an equivalent sequential one [20], it is often rather difficult to bring this latter to the fore (see e.g. the complicated sequential skeleton procedures described in [18]). Furthermore, in the geodesic case, sequential algorithms are far from being the most efficient ones: for example, the sequential computation of a geodesic distance function within a mask (see Fig. 6) requires an iteration of video and anti-video scannings until stability is reached. When the mask is not convex but "rolled-up", the number of image scannings involved may be important, as illustrated by Fig. 7. However, only the values of few pixels are actully modified after each scanning...

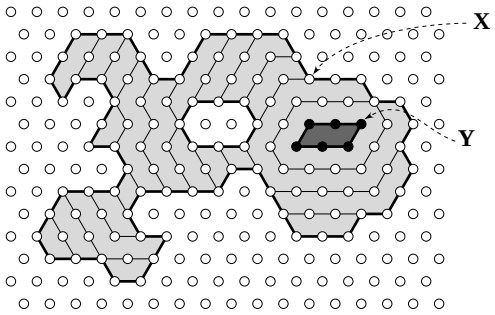

Figure 6 : *Level lines of a geodesic distance function within a set X: each pixel p of X is assigned the length of the shortest path between p and Y inside X.*

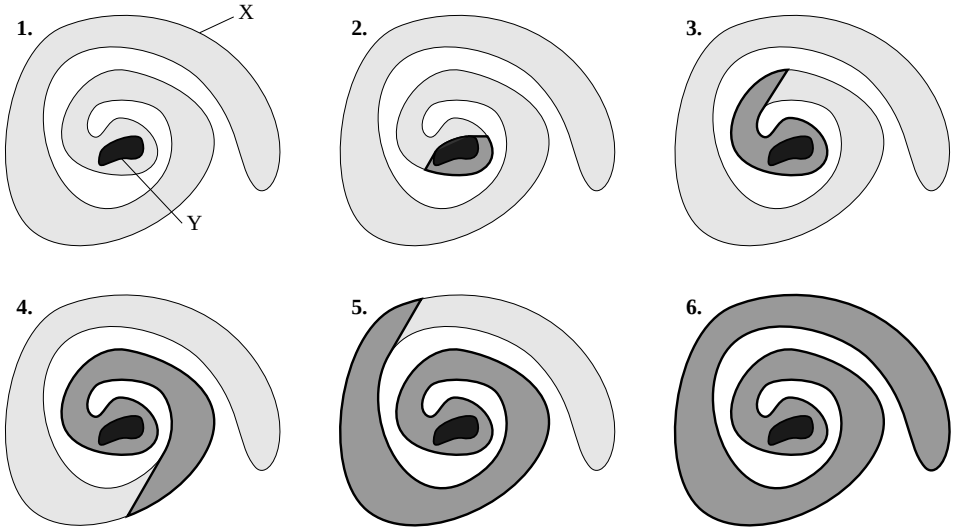

Figure 7 : *The sequential computation of geodesic distance functions in a rolled up mask may involve several complete image scannings: here, the hachured zones represent the pixels which are assigned a correct distance after each step.*

For these reasons, a further step in the design of efficient morphological algorithms consists in *considering only the pixels whose value may be modified.* A first scanning is used to detect the pixels which are the process initiators

*A hardware implementation of this algorithm has been realized and is described in [12].

and are typically located on the boundaries of the objects or regions of interest. Then, starting from these pixels, the information is propagated only in the relevant image parts. The algorithms described in the next two sections rely on these principles. They both require a *random access* to the image pixels as well as to the neighbors of a given pixel.

5 Algorithms Based on Loop and Chain Propagations

These methods have been proposed in 1988 by M. Schmitt [23] and are based on the following simple remark:

> *in a metric space (E, d), the boundary of the dilation $\delta(X)$ of a set X [25] by an isotropic structuring element is a curve which is parallel to the boundary of X.*

This is illustrated by Fig. 8. Hence, if one is able to determine quickly the curves parallel to a given one, the calculation of isotropic dilations can be efficiently performed. This process can then be used to compute a large number of other morphological transformations which can be defined from isotropic dilations in an incremental fashion. Among others, distance functions, which are nothing but "piles" of erosions, are attainable this way.

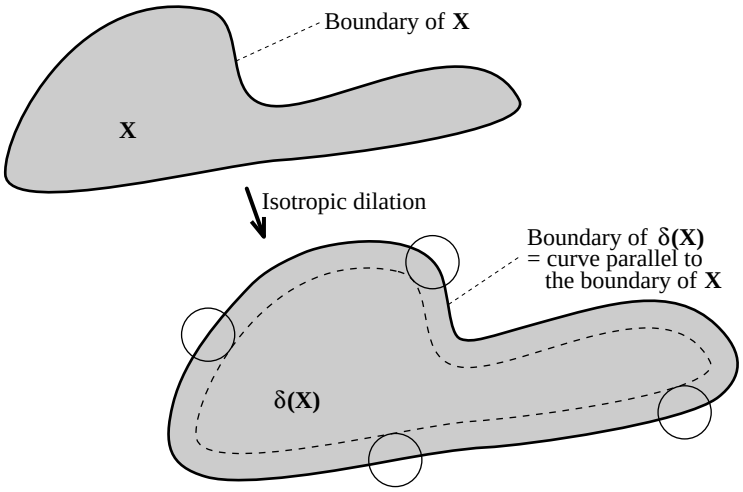

Figure 8: *The boundary of the dilated set $\delta(X)$ is parallel to the boundary of X.*

The first step of these algorithms consists therefore in a tracking of the contours of the image I under study and in their encoding as Freeman loops [8]. A loop L is a data structure made of:

1. an origin pixel Or_L,
2. a length $l(L)$,
3. a list of $l(L)$ integers of the segment $[0, 5]$, coding the elementary vectors $\vec{u}_0$, $\vec{u}_1$, ..., $\vec{u}_5$ of the hexagonal grid.

The two extremities of a loop coincide. An example of a loop and of its encoding is shown in Fig. 9.

Given a loop L coding the boundary of a set $X \subset \mathbb{Z}^2$, the dilated loop δL—coding the boundary of $\delta(X)$—is determined by means of *rewriting rules*. These rules allow one to derive from two successive contour elements of L a certain number (between 0 and 5) of contour elements of δL. The dilated loop is thus obtained from L in linear time with respect to $l(L)$. In the hexagonal case, there are exactly six rewriting rules (up to the six rotations), which are illustrated by Fig. 10. Rule number 4 may seem useless, but is in fact essential as soon as two successive dilations have been performed [23].

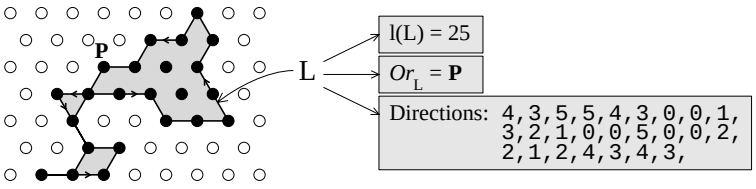

Figure 9 : *Example of a loop and of its encoding.*

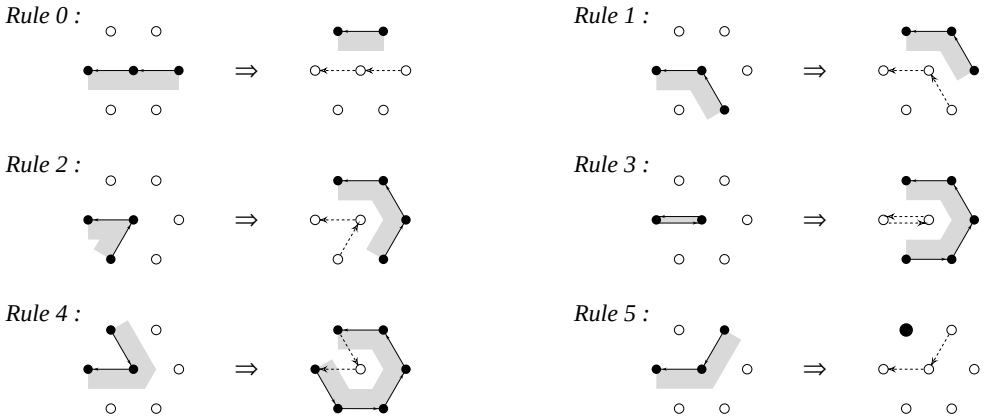

Figure 10 : *Rewriting rules allowing to determine δL from L.*

Now, once the dilated loops are determined, they must be written in the image and the corresponding pixels have to be given the appropriate value. For example in the case of a binary dilation, the pixels corresponding to the dilated loops have to be assigned value 1. In fact, while the loops are written in the original image, one can detect if some of them intersect. For example, it may well happen that two loops coming from two different connected components intersect after a dilation step, as illustrated by Fig. 11. In such cases, the overlapping parts are useless in the sequel and can be cut. Chains are thus created, which are nothing but loops whose extremities do not coincide. They are manipulated exactly as loops by using the rewriting rules of Fig. 10. They can well be cut again at further steps. An example of a chain dilation is shown in Fig. 12. The succession of operations described in this paragraph is referred to as *adjustment*. More precisely, during this adjustment step, one only keeps the chain or loop parts which are located in a given mask (set of pixels having a certain value) and give them the appropriate value.

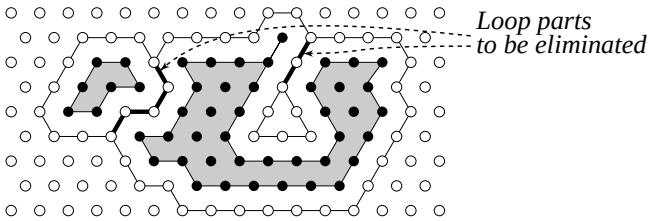

Figure 11 : *After dilation, certain loop parts must be eliminated. This is the adjustment step, in which* chains *are created.*

As an illustration, let us consider again the case of the hexagonal distance function. To determine it, we iterate dilations and adjustments of the chains until stability is reached. After each step, the value given to the "adjusted" chains is incremented by 1.

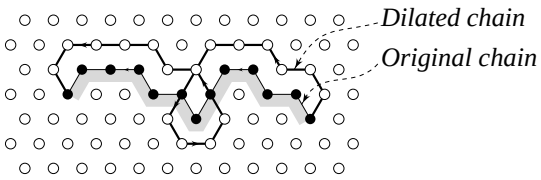

Figure 12 : *Dilation of a chain. Note that a loop has been created here, which will be eliminated during the adjustment step.*

Algorithm: distance function by chains and loops

- input: I, binary image; /⋆ distance directly computed in I ⋆/
- Assign value $+\infty$ to the frame of I;
- Track the contours of I^C (complement image) and encode them as loops;
- $dist \leftarrow 2$; /⋆ variable containing the current distance ⋆/
- Repeat until there remain chains or loops {
  Dilate the chains and the loops;
  Adjust them in the mask $\{p \in D_I \mid I(p) = 1\}$, giving the corresponding pixels value $dist$;
  $dist \leftarrow dist + 1$;
  }

Like almost all the algorithms relying on this chain propagation principle, the above one only requires two image scannings: one for the contour tracking step plus one scanning *of the feature pixels only* in the propagation step (in fact here, to avoid an additional scanning, the algorithm is designed to yield $\mathrm{dist}_I + 1$). They are thus extremely fast. Moreover, during the second step, loops and chains may as well be propagated inside a given mask: for this reason, the present methods are particularly suited to the computation of all binary geodesic transformations [23]. In addition, they can be adapted to better distances: the dodecagonal one can be reached by modifying the rewriting rules of Fig. 10 [23, pages 86–89] and it is even possible to adapt these algorithms to the actual Euclidean distance [34, 33]. Chain propagation algorithms also provide the only known efficient method for determining the *propagation function* of a particle [24, 16].

Nevertheless, the major drawback of this technique is that its adaptation to square grids is rather delicate, and its extension to n-dimensional spaces, impossible. In this respect, the algorithms based on queues of pixels which are discussed below are much more general.

6 Algorithms Based on Queues of Pixels

In this section, we again satisfy the principle according to which only the interesting image pixels are considered at each step. The image under study is regarded as a graph whose vertices are the pixels, and whose edges are provided by the discrete grid G. Then, instead of loops and chains, we make use of a *queue of pixels* to perform breadth-first scannings of this graph. This idea has already proved to be particularly interesting in image analysis [27] and morphology [31].

The queue is a First-In-First-Out data structure, which means that the pixels which are first put into it are those which can first be extracted. In other words, each new pixel included in the queue is put on one side whereas a pixel being removed is taken from the other side (see Fig. 13). In practice a queue is simply a large enough array of pointers to pixels, on which three operations may be performed:

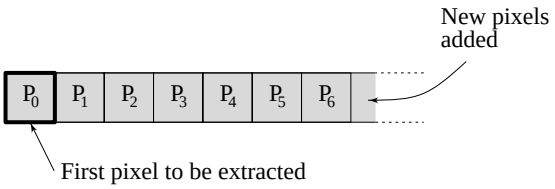

Figure 13 : *How a queue of pixels works.*

- *fifo_add(p):* *puts the (pointer to) pixel p into the queue.*
- *fifo_first():* *returns the (pointer to) pixel which is at the beginning of the queue, and removes it.*
- *fifo_empty():* *returns* true *if the queue is empty and* false *otherwise.*

The implementation of our distance function using this queue and the above operations is accomplished as follows:

Algorithm: distance function using a queue of pixels

- input: I, binary image;
  /⋆ The distance function is computed in I directly ⋆/
- For every pixel $p \in D_I$, do {
  /⋆ detection of the pixels to be initially put on the queue ⋆/
  If $I(p) = 1$ and $\exists p' \in N_G(p), I(p') = 0$ {
  fifo_add(p);
  $I(p) \leftarrow 2$;
  }
  }
- While *fifo_empty*$() = false$ {
  $p \leftarrow$ *fifo_first*();
  For every $p' \in N_G(p)$ {
  If $I(p') = 1$ {
  $I(p') \leftarrow I(p) + 1$;
  fifo_add(p');
  }
  }
  }

Here again, this algorithm actually yields $dist_I + 1$, a trick which avoids an additional image scanning.

Like the methods described in the previous section, queue based algorithms are extremely efficient, in both the Euclidean and the geodesic cases. The simplicity of the above distance function procedure is also remarkable, and this characteristic is shared by most FIFO algorithms. They are thus more suited than the chain propagation ones to the development of efficient procedures to compute complex transformations like skeletons and watersheds (see [31, chapters 7–8] and [32, 26]). In this latter case, the benefit provided by FIFO algorithms in terms of speed is considerable: a watershed transformation is determined in a couple of seconds whereas the same computation can take hours with parallel techniques! Additionally, the accuracy of these queue based algorithms was discussed in [26, 32] and proved to be higher than that of most other algorithms.

Moreover, and unlike the chain propagation algorithms, the present ones are extremely easy to adapt from one grid to another, since it suffices to modify the function to generate the neighbors of a given pixel. Similarly, their

extension to n-dimensional images and even to graphs is straightforward. They have been used to implement a vast number of morphological transformations on graphs [28], which have already been used in physical applications [11] and are expected to be of great interest for complex picture segmentation tasks [29].

7 Conclusion, Summary

Table 1 summarizes the qualities and drawbacks of the families of algorithms which have been briefly reviewed in this paper. Now, in practice, what algorithm should be chosen to implement a given transformation in a given environment? Of course, there is no absolute answer to this question. However, the following guidelines can be proposed:

- Parallel algorithms should generally be avoided, unless running on specialized architectures.
- Sequential algorithms constitute one of the best choices to implement grayscale reconstructions, grayscale dilations and erosions with some structuring elements, and granulometry functions.
- Chains and loops methods should be chosen whenever hexagonal grids and binary transformations are concerned. They are indeed the fastest in this case, particularly for geodesic operations (reconstruction, geodesic distance function, hole filling, labelling, etc), and provide the only known efficient propagation function algorithm.
- FIFO algorithms will be preferred in all other cases, and in particular in the following ones: geodesic transformations in square grids, n-dimensional or graph morphology, complex transformation like skeletons, SKIZ and watersheds.

Family of algorithms	speed	accuracy (if appropriate)	adaptation to other grids	development ease	hardware implementation
parallel	x	xx	xx	xxx	xxxx
sequential	xx	xx	xx	xx	xxx
loops and chains	xxxx	xxx	x	xxx	x
queues	xxx	xxx	xxxx	xxxx	x

Table 1 : *Respective qualities of the various families of algorithms.*

Clearly, the last two families constitute the choice of the future. Regardless from accuracy and flexibility considerations, chains and queues algorithms are often faster on conventional computers than parallel algorithms on specialized hardwares! Now, a few years after the introduction of the parallel morphological algorithms, the first specialized hardwares were build on these principles. Between the publication of the sequential distance function algorithm (1968) and its first hardware implementation (1989), more than twenty years have elapsed. It remains to hope that we will not have to wait twenty more years to see the first hardware realizing queue based morphological operations. Indeed, this would probably allow to compute complex morphological transformations in a couple hundredths of a second...

8 Acknowledgments

This work was supported in part by *Dassault Electronique* and the *National Science Foundation* under Grant MIPS–86–58150, with matching funds from *DEC* and *Xerox.* The author also wishes to thank Dan Friedman for his spontaneous participation.

9 References

1. S. Beucher and C. Lantuéjoul. Use of watersheds in contour detection. In *International Workshop on Image Processing, Real-Time Edge and Motion Detection/Estimation*, Rennes, France, 1979.

2. S. Beucher and L. Vincent. Introduction aux outils morphologiques de segmentation. In *Traitement d'Images en Microscopie à Balayage et en Microanalyse par Sonde Electronique*, pages F41–F43. ANRT, Paris, Mar. 1990.
3. G. Borgefors. Distance transformations in arbitrary dimensions. *Comp. Vis., Graphics and Image Processing*, 27:321–345, 1984.
4. G. Borgefors. Distance transformations in digital images. *Comp. Vis., Graphics and Image Processing*, 34:334–371, 1986.
5. L. Calabi and W. Harnett. Shape recognition, prairie fires, convex deficiencies and skeletons. Technical Report 1, Parke Math. Lab. Inc., One River Road, Carlisle MA, 1966.
6. P. Danielsson. Euclidean distance mapping. *Comp. Graphics and Image Processing*, 14:227–248, 1980.
7. H. Digabel and C. Lantuéjoul. Iterative algorithms. In J.-L. Chermant, editor, *2nd European Symposium on Quantitative Analysis of Microstructures in Material Science, Biology and Medicine*, pages 85–99, Stuttgart FRG, 1978. Riederer Verlag.
8. H. Freeman. On the encoding of arbitrary geometric configurations. *IEEE Transactions on Computers*, C10:260–268, 1961.
9. S. Gesbert, V. Howard, D. Jeulin, and F. Meyer. The use of basic morphological operations for 3-d biological image analysis. *Transactions of the Royal Microscopical Society*, 1, 1991.
10. M. Golay. Hexagonal pattern transforms. *IEEE Trans. on Computers*, 18(8), 1969.
11. D. Jeulin, L. Vincent, and G. Serpe. Propagation algorithms on graphs for physical applications. *JVCIR*, 3(2):161–181, June 1992.
12. J.-C. Klein and R. Peyrard. PIMM1, an image processing ASIC based on mathematical morphology. In *IEEE's ASIC Seminar and Exhibit*, pages 25–28, Rochester NY, 1989.
13. C. Lantuéjoul. Issues of digital image processing. In R. M. Haralick and J.-C. Simon, editors, *Skeletonization in Quantitative Metallography*. Sijthoff and Noordhoff, Groningen, The Netherlands, 1980.
14. C. Lantuéjoul and F. Maisonneuve. Geodesic methods in quantitative image analysis. *Pattern Recognition*, 17(2):177–187, 1984.
15. B. Laÿ. Recursive algorithms in mathematical morphology. In *Acta Stereologica Vol. 6/III*, pages 691–696, Caen, France, Sept. 1987. 7th International Congress For Stereology.
16. F. Maisonneuve and M. Schmitt. An efficient algorithm to compute the hexagonal and dodecagonal propagation function. In *5th European Congress For Stereology*, pages 515–520, Freiburg im Breisgau FRG, Sept. 1989. Acta Stereologica. Vol. 8/2.
17. G. Matheron. *Random Sets and Integral Geometry*. John Wiley and Sons, New York, 1975.
18. F. Meyer. Skeletons and perceptual graphs. *Signal Processing*, 16(4):335–363, 1989.
19. F. Preparata and M. Shamos. *Computational Geometry: an Introduction*. Springer Verlag, 1985.
20. A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *J. Assoc. Comp. Mach.*, 13(4):471–494, 1966.
21. A. Rosenfeld and J. Pfaltz. Distance functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.
22. H. Samet. The quadtree and other related hierarchical data structures. *ACM Computing Surveys*, 16(2):87–260, 1984.
23. M. Schmitt. *Des Algorithmes Morphologiques à l'Intelligence Artificielle*. PhD thesis, Ecole des Mines, Paris, Feb. 1989.
24. M. Schmitt. Geodesic arcs in non-euclidean metrics: Application to the propagation function. *Revue d'Intelligence Artificielle*, 3 (2):43–76, 1989.
25. J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
26. P. Soille and L. Vincent. Determining watersheds in digital pictures via flooding simulations. In *SPIE Vol. 1360, Visual Communications and Image Processing*, pages 240–250, Lausanne, Switzerland, 1990.

27. L. van Vliet and B. J. Verwer. A contour processing method for fast binary neighbourhood operations. *Pattern Recognition Letters*, 7:27–36, Jan. 1988.

28. L. Vincent. Mathematical morphology on graphs. In *SPIE Vol. 1001, Visual Communications and Image Processing*, pages 95–105, Cambridge, MA, Nov. 1988.

29. L. Vincent. Mathematical morphology for graphs applied to image description and segmentation. In *Electronic Imaging West 89*, pages Vol. 1, pp. 313–318, Pasadena, CA, 1989.

30. L. Vincent. Morphological shading and shadowing algorithm. In Hermès, editor, *PIXIM 89, Computer Graphics in Paris*, pages 109–124, Paris, 1989.

31. L. Vincent. *Algorithmes Morphologiques à Base de Files d'Attente et de Lacets: Extension aux Graphes.* PhD thesis, Ecole des Mines, Paris, May 1990.

32. L. Vincent. Efficient computation of various types of skeletons. In *SPIE Vol. 1445, Medical Imaging V*, pages 297–311, San Jose, CA, 1991.

33. L. Vincent. Exact euclidean distance function by chain propagations. In *IEEE Int. Computer Vision and Pattern Recog. Conference*, pages 520–525, Maui, HI, June 1991.

34. L. Vincent. Morphological transformations of binary images with arbitrary structuring elements. *Signal Processing*, 22(1):3–23, Jan. 1991.

35. H. Yamada. Complete euclidean distance transformation by parallel operations. In *7th International Conference on Pattern Recognition*, pages 69–71, Montreal, 1984.