ELSEVIER

Pattern Recognition Letters 17 (1996) 1451-1460

Pattern Recognition Letters

Fast computation of morphological operations with arbitrary structuring elements

Marc Van Droogenbroeck[a], Hugues Talbot *

[a] *Belgacom New Developments, Bd. Emile Jacqmain 177, B-1210 Brussels, Belgium*

[b] *CSIRO - Mathematical and Information Sciences, Locked Bag 17, North Ryde, NSW 2113 Australia*

Received 12 December 1995; revised 9 September 1996

Abstract

This paper presents a general algorithm that performs basic mathematical morphology operations, like erosions and openings, with *any arbitrary shaped* structuring element in an efficient way. It is shown that our algorithm has a lower or equal complexity but better computing time than all comparable known methods.

Keywords: Mathematical morphology; Algorithm; Arbitrary shape

1. Introduction

Mathematical morphology, denoted MM hereafter, is a theory devised for the shape analysis of objects and functions (Serra, 1982). Usual morphological operators are made of two parts: (i) a reference shape, called a structuring element or function, that is translated and compared to the original function all over the plane, and (ii) a mechanism that details how to carry out the comparison. Structuring element will be abbreviated as SE hereafter.

MM has become increasingly popular over the last few years. Part of this success is due to the remarkable increase in efficiency of many of the MM algorithms. Many MM operations which used to require expensive dedicated hardware to run in reasonable times can now be performed on standard workstation or even personal computers. In this paper, we present such an algorithm well-suited for basic morphological operations with large arbitrary shaped structuring elements.

Definitions and notations. We briefly recall the definitions and notations used in this paper. We use the Minkowsky addition definitions as used in (Haralick et al., 1987). Let f be a function defined on the Euclidean space $\mathbb{Z}^2$ ($f : \mathbb{Z}^2 \to \mathbb{Z}$), $B \subseteq \mathbb{Z}^2$ a flat structuring element and $f_b(x) = f(x - b)$ the translation of $f(x)$ by b in $\mathbb{Z}^2$. The erosion $f \ominus B$ (respectively dilation $f \oplus B$) of a function f by B maps f into another function defined as

$$(f \ominus B)(x) = \min_{b \in B} \{f_{-b}(x)\}, \tag{1}$$

$$(f \oplus B)(x) = \max_{b \in B} \{f_b(x)\}, \tag{2}$$

where min and max denote respectively the minimum and the maximum operations.

These past years, considerable effort has been put into the development of fast and efficient algorithms

* Corresponding author. E-mail: hughues.talbot@cmis.csiro.au.

PII S0167-8655(96)00113-4

for complex MM operations on simple desktop workstations. Algorithms for basic operations like the erosion and dilation with a large SE are also under discussion in the literature, mainly because these operations belong to most industrial procedures based on MM. The aim is to reduce the number of steps that would be required in a direct implementation of definitions (1) and (2). The simplest implementation, called the *trivial method* in this article, consists in searching for each point p the minimum value (in the case of an erosion) among all the values $f(p+b)$ where $b \in B$. This method is acceptable for small SEs, but leads to very long execution times for big SEs. A simple example illustrates the inherent redundancy of the trivial method. Suppose $B = \{a, b\}$ and $q = p - a + b$. Then

$$(f \ominus B)(p) = \min\{f(p+a), f(p+b)\} \quad \text{and}$$
$$(f \ominus B)(q) = \min\{f(p+b), f(q+b)\}.$$

$f(p+b)$ appears in both expressions but should, in an ideal situation, only be referred to once. Larger SEs lead to even more duplications. Until very recently, no other solution than the *trivial method* was available for operations with SEs other than squares or hexagons.

Commercial softwares provide the possibility to erode with a square of size $[(2n+1) \times (2n+1)]$ that is obtained by repeating n erosions with a $[3 \times 3]$ square. This method is referred to as the *linear decomposition method*. Originated by Pecht (1985) and enhanced by Van den Boomgaard (1992), the *logarithmic decomposition* improves the linear decomposition by removing some redundant computations. However, the principle cannot be extended to arbitrary shaped structuring element.

In the restricted case of binary MM, Vincent (1991) proposed an efficient solution that considers only the pixels of the edge of the binary objects, and then moves along this contour pixel by pixel, writing on the output image only those pixels of the SE which could not be reached at the precedent move (the non-overlapping pixels), or the entire SE at the beginning. This allows the coding of an efficient dilation. The erosion was obtained by dilating the complementary image. Still in the binary case, Liang and Wong (1993) proposed an equivalent solution to the problem using a hierarchical pyramid data structure.

The problem is more difficult when dealing with gray-level functions. Meyer (1990) proposed a better solution than the trivial approach, based on hierarchical queues, which also took advantage of the overlapping parts of SEs when considering neighboring points. Gratin et al. (1993) proved that the Meyer method indeed could be used with flat SEs of any shape. They proposed to replace Meyer's hierarchical queues with a simple sorting of all the pixels of the image, which proved significantly faster. It was shown in the same paper that the complexity of the Gratin-Meyer algorithm was the same as the complexity of the trivial method (i.e. linear with the area of the structuring element), but that it was up to 50 times faster depending on the structuring element and the content of the image.

The key idea in these approaches is that, in most cases, it is redundant to completely re-compute a minimum for all possible translations vectors $b \in B$ when the erosion of two neighboring points of the image is to be determined. Indeed, for two such points, the sets on which the minimum must be computed differ only by a contour which is one pixel wide, regardless the need for a hierarchical or sorting approach to the problem, as with Meyer's method. Many authors have applied this idea of a "sliding window" in order to compute rank-order filters (see (Chaudhuri, 1990; Gil and Werman, 1993; Huang et al., 1979)). However, they have only proposed methods for rank filters on rectangular windows. In the field of MM, the *shape* of the SE is extremely important.

In the next section, we extend the sliding window principle to SEs of any shape, and show that this method leads to a more efficient and faster algorithm than the Gratin-Meyer algorithm.

2. Algorithm description

2.1. Basic principle

In the following, for the sake of simplicity, we will only present the case of erosion with flat structuring elements. The case of the dilation is obtained by duality, and the case of non-flat structuring elements is discussed in Section 3. It is also assumed that B contains the origin for simplicity.

In order to determine $(f \ominus B)(p)$, the minimum of f inside the "window" B_p has to be known. This computation must take place for all the pixels of f,

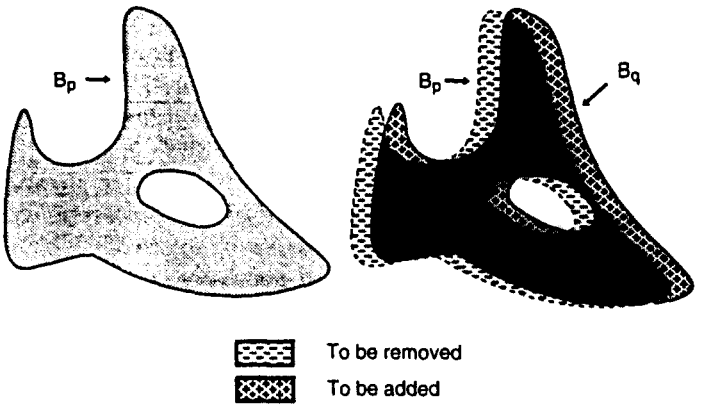


Fig. 1. Effects of a small translation of the structuring element during an image scan. The gray area of the right-hand figure represents locations of values both considered for $(f \ominus B)(p)$ and $(f \ominus B)(q)$.

and can be done using a regular scanning from pixel to pixel. For example on a random line of f, the scanning can be done from left to right and corresponds to a small translation of B in the same direction. This is the situation depicted in Fig. 1, where B_p is a given position of the SE and B_q is the next translated SE. The neighboring value $(f \ominus B)(q)$ is the minimum of f inside B_q after the translation $q-p$. From $(f \ominus B)(p)$, the values of f inside $B_p \cap B_q$ are already known. To obtain the minimum inside B_q, we therefore only need to consider the values of f whose locations are filled with '−' and '+' symbols in Fig. 1. If $q-p$ is small compared to the size of B, the parts to be removed from or to be added to B_q are also small. In the digital case, where $q-p$ is one pixel long, these parts are only one pixel thick and correspond to a subset of the contour of B. From the computer's point of view, these parts can be efficiently accessed, and the cost to compute the minimum inside B_q is only a fraction of what it would have been if we had not taken advantage of our knowledge of f inside B_p. A similar local adaptation is done after each translation.

To be able to compute the minimum of f inside B_q, it is sufficient to memorize only the gray-level *histogram* of all the points of f inside B_p, then to count the points which appear and those which disappear *in the histogram* during the translation towards B_q. The minimum value can be obtained from the new gray-level histogram, which is indeed the histogram of the values of f inside B_q. It is a lot more efficient to work with the histogram than to search the minimum value after each translation.

2.2. Algorithm complexity

As the time needed to compute the minimum value in a histogram does not depend upon the number of points which form this histogram, but depends only on the number of possible grey-levels in the image, we can derive the theoretical complexity of the proposed algorithm.

2.2.1. Worst-case scenario

The worst-case scenario happens when the SE being used is such that there is no intersection between the SE at a given position and its subsequent position on the grid after only one translation. In this case, the algorithm will examine all the points in the SE for every move. In other words, the complexity of the algorithm in this case is the same as for the trivial method, which is proportional to the area A of the SE.

Examples of such structuring elements are lines segments not oriented in the translation direction, or one pixel thick SE (thin rings).

2.2.2. Best-case scenario

The best-case scenario happens when the number of pixels which are modified during a translation step is very small compared with the total number of pixel of the SE. In this case, for a granulometry made of such SE (Matheron, 1975), the complexity of the proposed method is constant or almost constant.

This is in particular the case for line segments orientated in the propagation direction: the complexity of the proposed algorithm and the computing time are

independent of the length of the segment.

2.2.3. Typical scenario

Typically, SE have a perimeter much smaller than their area. This is in general increasingly true with the size of the SE. For such a typical SE, the set of pixels modified during one translation step of the SE is included in the perimeter of the SE, which is asymptotically proportional to the square root of the Ferret diameter of the SE.

For example, if we consider a square SE, the pixels modified during one translation step are those along the edge of the SE perpendicular to the translation direction, as in Fig. 2. If A is the area of the SE, then the complexity of the method is in $O(\sqrt{A})$.

2.3. Implementation issues

In the digital case, we need to be able to deal with one pixel long translations of the considered SE along a subset of the principal directions of the underlying grid; examples of subsets are given in Fig. 3.

We also need to start the process by calculating once (and, if possible, only once) the complete histogram under one position of the SE (for example the topmost, leftmost position of the SE), and then proceed with size-1 translations of the SE following the methods described in the previous section, in order to cover the entire image.

We know it is sufficient to deal with the points which are included in the parts to be added or removed during size-1 translations of the SE. We can call these points *critical points*. The critical points depend on the shape of the SE and on the direction of translation. There are 3 translation directions in the case of the square grid (0, $-\pi/2$ and π) and 4 in the case of the hexagonal grid (0, $-\pi/3$, π and $-2\pi/3$). These points are included in the contour of the SE and are therefore few in number for most SEs.

During scanning, and in the case of pure MM operators (other rank order filters excluded), we shall keep a variable which holds the position in the histogram we are interested in: minimum in the case of an erosion, maximum in the case of the dilation. This value is read once per translation (that is the "output" value) and is in practice seldom modified. Although this notion is difficult to demonstrate, the argument goes thus: this relevant histogram value changes only when a pixel with a lower (erosion) or higher (dilation) value is introduced in the histogram, in which case the modification is immediate, or when the corresponding frequency in the histogram comes to 0, indicating that there is no longer a pixel under the SE which holds this value, in which case a scanning of the histogram is necessary. This approach eliminates the necessity to actually look into the histogram after each translation. Finally, even in the case of a very noisy image where the extremal value would change often, the scanning of the histogram occurs in constant time and this operation does not affect the complexity of the algorithm.

Before starting the operation, an analysis of the SE is necessary to find the critical points on the contour of the SE. In practice we can hold these points in arrays of 2D vectors, these vectors being the position of the critical points relative to the origin of the SE. One array will contain the position of the points to be removed, another the position of the points to be added. These easily accessed arrays are filled only once at the beginning of the algorithm, and are only consulted afterwards. The analysis of the SE can be done by effectively making the relevant translations and finding the points that change with logical operations. This allows to process easily the complex cases (SEs with holes, non-connected SEs, etc.).

It is also critical to process correctly the edges of the image. The best practice is to add edges of sufficient size to the original image. The size of the edges are easy to determine: they are equivalent to those of the bounding box of the SE, as in Fig. 4.

2.4. Algorithm implementation

Here is now the complete pseudo-code of the algorithm in the case of an erosion by a flat SE on a square grid.

Algorithm (*Erosion by a structuring element B*).

- **Declaration of data**
  - *imIn*, the original image f
  - *imOut*, the output image
  - B, the binary image containing the SE
  - *histo*, the histogram
- **Initialization**
  Finding of the border points of B in each direction

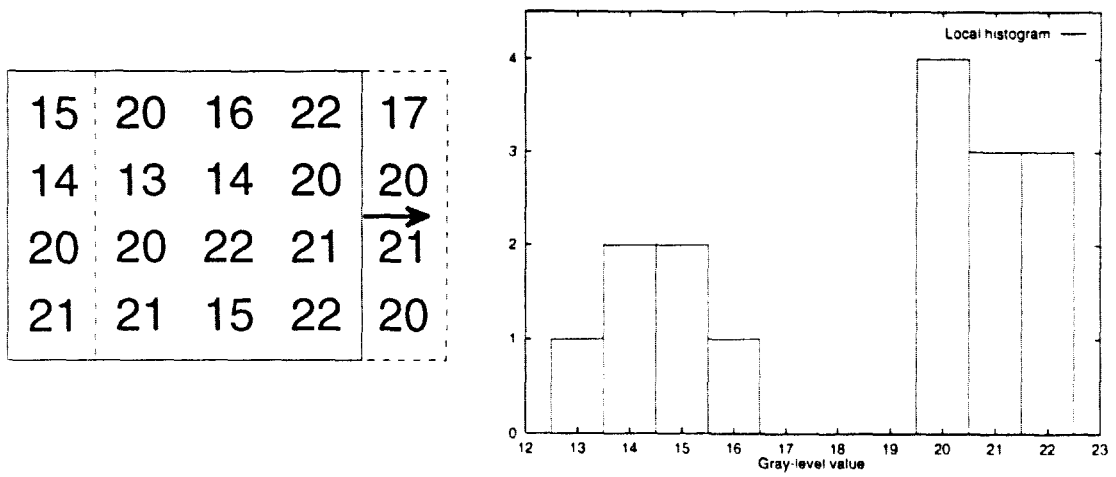
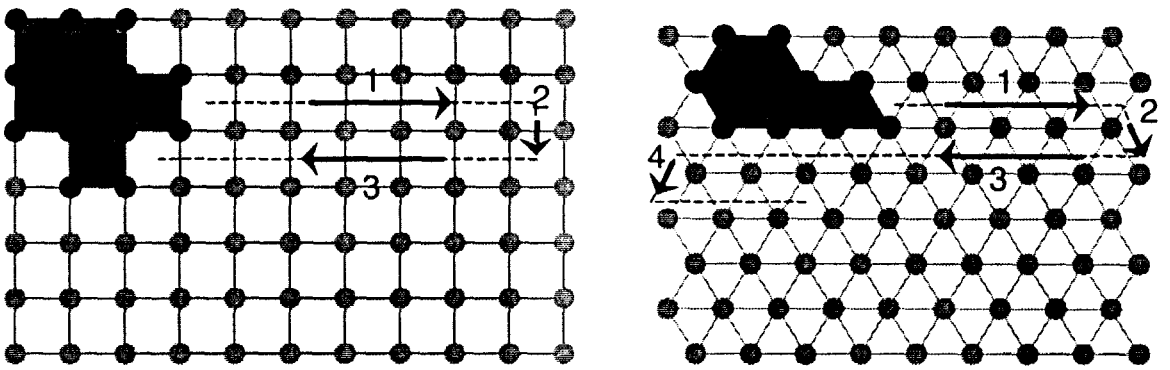


Fig. 2. A different local histogram is associated to each location of the window. The one pixel to the right translation of the square will affect the minimum given in the right-hand histogram.



(a) *Scanning a square grid* (b) *Scanning an hexagonal grid*

Fig. 3. Proposed translations on a digital grid.

using translations
Filling of the histogram for the first position of B

- **Main loop**

For each even line l of *imIn*
For each point c of line l
Translate B one pixel to the right and adapt the histogram ;
$imOut[c, l] \leftarrow \min(histo)$;
EndFor
Translate B one pixel to the bottom and adapt the histogram;
For each point c of line $l + 1$
Translate B one pixel to the left and adapt the histogram;
$imOut[c, l + 1] \leftarrow \min(histo)$;
EndFor
EndFor

To implement a dilation, it is sufficient to replace the min( ) operator by a max( ). Fig. 5 represents the result of a numerical opening, that is an erosion followed by a dilation, by a structuring element resembling a face, of the famous "Mona Lisa" painting. This allows to find the location of the face of Mona Lisa on the painting.

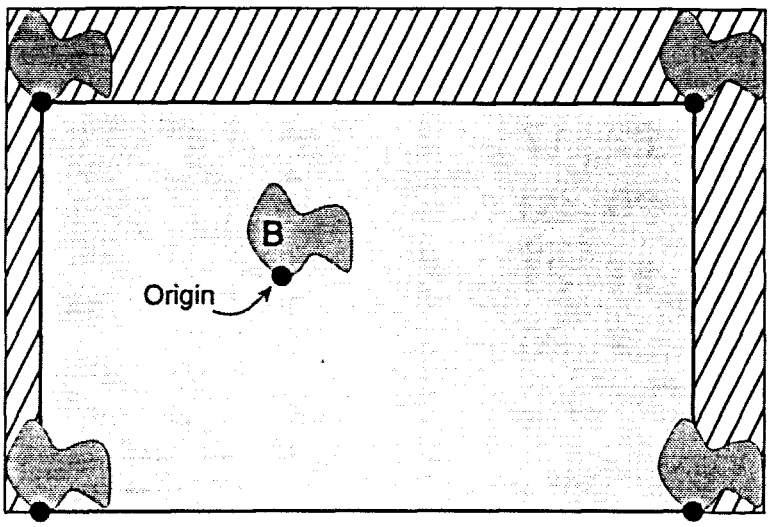


Fig. 4. Additional edge to be added to the original image during the operation.

(a) *Original image* (b) *Structuring element* (c) *Final result*

Fig. 5. Pattern recognition using an opening with a face-like structuring element.

2.5. *Results and discussion*

Let us now consider the computing time on a sample image. We have considered a 512 × 512 square grid image on which a series of erosions have been performed. The structuring element was a standard square so that the methods can be compared with the linear decomposition method.

The following methods were compared:

1. The *trivial method*, consisting in a comparison of all the pixels of the input image under the SE for all the positions of the SE.
2. The *Gratin–Meyer method* which consists in a hierarchical approach.
3. The proposed method presented above and men-

tioned as *Talbot–Van Droogenbroeck* on the graph.

4. And a reference technique: the *linear decomposition method* that erodes an image by an $[N \times N]$ (N must be odd) square by repeating $(N-1)/2$ erosions by a $[3 \times 3]$ square.

Fig. 6 presents the results in a log-log scale graph. Computing times were obtained on a NeXT 68040 25 MHz workstation (equivalent to a Sun SparcStation I) with the `gcc` compiler and the `gprof` profiler.

We can see with this result that the *trivial method* yields extremely long computing times. For an erosion with a $[61 \times 61]$ square, the computing time is close to 45 minutes.

The *linear decomposition method* yields the best result, albeit closely followed by the proposed method. The key point here is that the decomposition method can be used only with a limited number of SEs, like squares in this case, whereas the proposed method can be used with arbitrary SE.

By construction, the decomposition method has a complexity in $\sqrt{A}$, A being the number of pixels in the SE. Indeed, the number of loops with the unit square SE needed to compute an erosion with a square SE containing A pixel is proportional to the length of the edge of such a square, which is equal to $\sqrt{A}$. The fact that our method follows so closely the decomposition method in terms of computing time shows that the complexity estimate in $\sqrt{A}$ for our method was correct.

The *Gratin–Meyer method* gives much better computing times than the trivial method, but this method is inferior to the proposed method. The *Gratin–Meyer method* is always slower than the proposed method, and because of the lower complexity of the proposed method, the difference in computing time increases quickly with the size of the SE.

It must be noted that the comparison between the *Gratin–Meyer method* and the proposed method extends to shapes more complex than simple squares. For example, nearly identical results are obtained when using a series of scaled SEs similar to the one used in Fig. 5.

3. Extensions

It is possible to extend the presented algorithm in at least four different ways.

3.1. Non-flat structuring elements

Suppose that function $g(x) : \mathbb{Z}^2 \rightarrow \mathbb{Z}$ only takes non-zero values on D. The gray-scale erosion and dilation of f by a non-flat structuring element g are respectively defined by

$$(f \ominus g)(x) = \min_{d \in D}\{f(x+d) - g(d)\}, \tag{3}$$

$$(f \oplus g)(x) = \max_{d \in D}\{f(x-d) + g(d)\}. \tag{4}$$

A non-flat SE (so-called 3D SE for 2D images) is like a structuring function non-necessarily constant on its definition interval. It should not be mistaken with 3D SEs for 3D images. On digital images, computing a basic MM operation with such an SE is equivalent to computing a weighted extremum.

The method presented in this paper applies equally well to non-flat SEs. The same principle applies: points in the SE that do not change during a one-pixel translation do not need to be re-examined during this very translation. However, in the general case, the set of points which do change during one-pixel translations is not a subset of the contour of the SE: depending on the geometry of the SE, some points within the SE might need to be considered as well.

In a more specific manner, if during the translation of the SE from a position to an adjacent one, the weight of a given pixel on the image f remains the same, then it is not necessary to update the gray-level histogram under the SE for that position. If, on the other hand, the weight does vary, then we must consider this position much in the same way contour pixels were considered in the flat case.

3.1.1. Algorithm complexity

Flat structuring elements are actually a subset of non-flat structuring elements. The same discussion presented in Section 2.2 also applies to non-flat structuring elements.

In the common case of real non-flat SEs, a large proportion of the SE needs to be updated for each translation. This proportion is usually independent of the size of the SE. This means that the complexity of the method in that case will be the same as of the trivial method. However, since not all points need to be updated, contrary to the trivial method, a substantial gain in computing time can be expected, as shown in

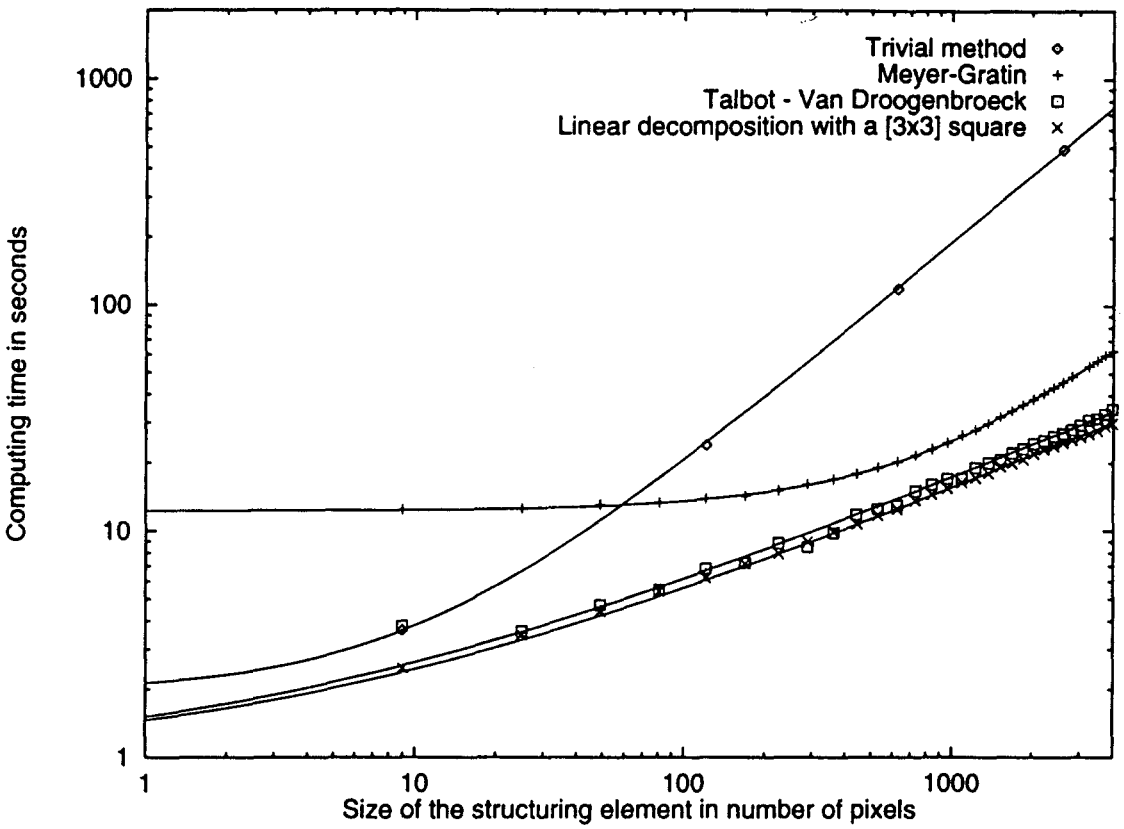


Fig. 6. Computing time. Erosion by squares on a [512 × 512] square grid image (log-log scale).

Table 1
Comparing computing time (in seconds) for the proposed method versus the trivial method in the case of a rolling ball SE

Radius of the SE (pixel)	Proposed method (sec)	Trivial method (sec)
5	11	13
10	28	50
15	70	128
20	140	230
30	330	557

Table 1. In this table, computing time are shown for both the proposed method and the trivial method in the case of the erosion of the image of Mona Lisa (Fig. 5) by the very common spherical "rolling ball" structuring element (i.e., a circular SE with weights arranged as to describe a hemisphere in grey-level). In this case, which is quite typical, a computing time gain of nearly 50% can be obtained by using the proposed method.

3.1.2. *Implementation*

The implementation of the algorithm for 3D SE is almost the same as for flat SE. The only part which changes is the necessity to apply the corresponding weights to the points that enter and leave the histogram calculation. In practice, this management takes a negligible amount of computing time.

It is possible to actually implement the case of the flat SE as a subset of the general case. As a result, the total computing time is directly proportional to the number of points which need to be updated with each translation of the SE, regardless of whether the SE is flat or not, as shown in Fig. 7.

3.2. *3D images*

The same method described above applies to 3D images. One needs to consider a larger number of scanning directions (in the case of the cubic lattice, for

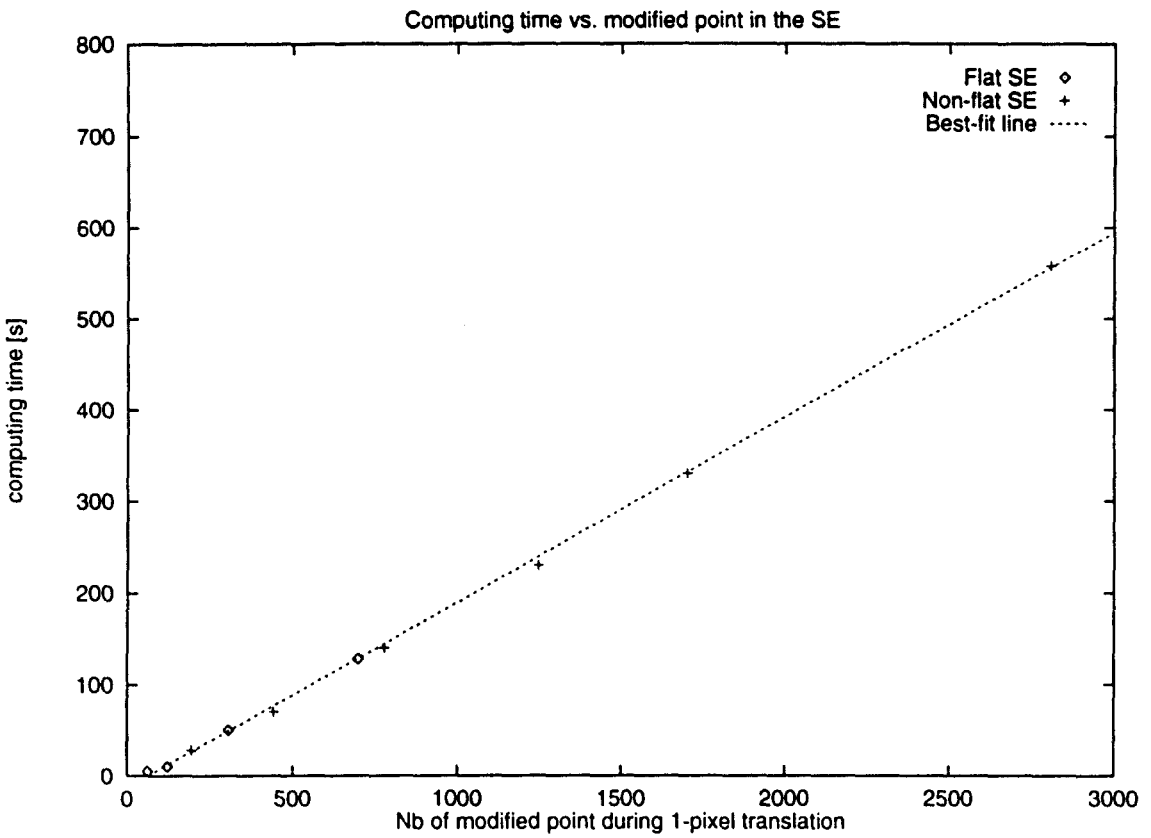


Fig. 7. Computing times for erosions by various SE on a given image. Both flat and non-flat SE are handled in the same way. The computing time is proportional to the number of points which need to be updated at each translation.

example, we need to consider 5 scanning directions).

The set of critical positions in the SE will have the dimension of a unit thick surface.

3.3. Rank order filters

It is also possible to realize rank filters on non-rectangular windows with this method. Instead of looking for a minimum or a maximum in a window, we just need to look for the median value (for example) of the gray-level histogram of the points of the image f under the SE B. Since this histogram is readily available for all the positions of B, this extension comes at no extra cost.

3.4. Optimization of the direction of translation

The direction in which the translation of the SE is made makes no difference to the final result. However, the directions in which the SE are translated can make a difference in terms of computing speed, depending on the SE's shape. It is better to choose a translation direction such that the width of the SE in the perpendicular direction is minimal, if such a direction exists. This makes a dramatic difference for line SE for example.

The actual translation can be achieved either by a prior rotation of the image (which is exact only in a limited number of directions), or by using actual translation of the SE, along either Bresenham or periodic lines, as defined by Soille et al. (1996).

4. Conclusions

A new method allowing the computation of basic morphological operations with *arbitrary* flat structuring elements on 2D gray-level images has been pre-

sented.

It was shown that this method is less complex, more efficient and faster than all existing methods for the computation of such operations.

Extensions of this method to non-flat structuring elements, 3D images and rank-order filters are also possible.

Besides traditional application of morphological operators like pattern matching with arbitrary shapes or filtering with isotropic structuring elements, this method has also been used to compute isotropic gray-level granulometries in reasonable computing times.

References

Chaudhuri, B.B. (1990). An efficient algorithm for running window pel gray level ranking in 2-D images. *Pattern Recognition Letters* 11 (2), 77–80.

Gil, J. and M. Werman (1993). Computing 2-d min, median and max filters. *IEEE Trans. Pattern Anal. Mach. Intell.* 15 (5), 504–507.

Gratin, C., F. Meyer and H. Talbot (1993). Fast gray-level morphological transforms with any structuring elements. In: *Visual Communications and Image Processing '93*, Boston. SPIE.

Haralick, R., S. Sternberg and X. Zhuang (1987). Image analysis using mathematical morphology. *IEEE Trans. Pattern Anal. Mach. Intell.* 9 (4), 532–550.

Huang, T., G. Yang and G. Tang (1979). A fast two-dimensional median filtering algorithm. *IEEE Trans. Acoust. Speech Signal Process.* 27 (1).

Liang, E. and E. Wong (1993). Hierarchical algorithms for morphological image processing. *Pattern Recognition* 26 (4), 511–529.

Matheron, G. (1975). *Random Sets and Integral Geometry*. Wiley, New York.

Meyer, F. (1990). Un algorithme ordonné de dilatation. Internal report, Ecole des Mines de Paris, Centre de Morphologie Mathématique.

Pecht, J. (1985). Speeding up successive Minkowski operations. *Pattern Recognition Letters* 3 (2), 113–117.

Serra, J. (1982). *Image Analysis and Mathematical Morphology*. Academic Press, New York.

Soille, P., E.J. Breen and R. Jones (1996). Recursive implementation of erosions and dilations along discrete lines at arbitrary angles. *IEEE Trans. Pattern Anal. Mach. Intell.* 18 (5), 562–566.

Van den Boomgaard, R. (1992). Mathematical morphology: Extensions towards computer vision. PhD thesis, Amsterdam University.

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