ON THE IMPLEMENTATION OF MORPHOLOGICAL OPERATIONS

MARC VAN DROOGENBROECK
Belgacom, New Developments
121, Boulevard E. Jacqmain
B-1210 Brussels, Belgium
Fax: (+32) 2 202 84 52

Abstract.
This work presents different novel approaches to the implementation of morphological operations, like opening and close-open filter. In the restricted case of binary mathematical morphology, we propose a propagation algorithm well-suited for opening. Another algorithm, based on a "sliding window", is described for the implementation of grey-level openings. Finally we present some theoretical results that simplify the computation of complex operations.

Key words: Fast algorithm, Filter, Histogram, Opening

1. Introduction

Over the years, mathematical morphology has become very attractive. The gain of popularity is due to the recent development of powerful tools, like the watershed, but also to the discovery of fast algorithms that make morphology competitive in comparison with linear operations.

The fast implementation of morphological operations is an important task for both hardware and software. Unfortunately, the requirements of these two are often in conflict. In this paper, we propose a way of optimizing the implementation of morphological operations. The leading idea is that *knowledge about previous operations can be used to decrease the complexity of further morphological stages*, a property which could for instance significantly decrease the computation time of alternate sequential filters. In fact, it has been thought for many years that an opening is always computed as an erosion followed by a dilation. This is not true as will appear from the algorithms described in sections 3 and 4.1.

2. Reminder

We briefly recall the definitions and notations used in this paper. Let f be a function defined on $\mathbf{R}^2$ and $B \subseteq \mathbf{R}^2$ a planar structuring element. The erosion $f \ominus B$ (resp. dilation $f \oplus B$) of a function f by B transforms f into another function defined as:

$$f \ominus B(x) = \bigwedge_{b \in B} f_{-b}(x) = \bigwedge_{b \in B} f(x+b)$$

$$f \oplus B(x) = \bigvee_{b \in B} f_b(x) = \bigvee_{b \in B} f(x-b)$$

for every $x \in \mathbf{R}^2$.

The opening $f \circ B$ and closing $f \bullet B$ result from cascading erosion and dilation operations:

$$f \circ B = (f \ominus B) \oplus B$$
$$f \bullet B = (f \oplus B) \ominus B$$

By definition, the opening of a set X with respect to a structuring element B, denoted $X \circ B$, is the union of the translates of B included in X. It can also be obtained from the above definition by replacing the infimum and the supremum respectively with the intersection and the union.

3. Propagation algorithm for binary opening

As could be assumed from the definition, the computation of the opening is proportional to the surface area. In practice, testing the correspondence for each position would be too slow and several methods were proposed to overcome this difficulty:

- *Linear decomposition.* A large structuring element is decomposed into a succession of dilations of smaller sets. Suppose for example that $C = B \oplus B$ then $X \circ C = (((X \ominus B) \ominus B) \oplus B) \oplus B$;
- *Logarithmic decomposition.* Originated by PECHT [5], this method enhances the linear decomposition by removing some redundant computations ;
- SCHMITT [6] and VINCENT [9] proposed methods which analyze the border of both X and B.

Other techniques were also suggested in the literature (cf. [1, 8]). All these methods perform an opening as an erosion followed by a dilation. The algorithm described hereafter is based on the propagation of the structuring element inside of X and will not proceed by intermediate erosions and dilations.

3.1. ALGORITHM DESCRIPTION

The proposed algorithm is adaptive in the sense that it relies locally on the set X: an early step detects the border elements of X where B could be anchored and propagates the structuring element inside the set in order to minimize the number of tests for each displacement.

In mathematical terms, the idea is the following. Let B_p, B_q denote the translation of B by p, q, and $\setminus$ the set subtraction. Suppose $B_p \subseteq X$. If $B_q \setminus B_p \subseteq X$ then $B_q \subseteq X \circ B$. The algorithm proceeds in two steps: (1) detection of B_p-like candidates and (2) check if $B_q \setminus B_p \subseteq X$ for several positions q close to p. Ideally we could expect the number of tests to be small: 1 per element added to $X \circ B$. This lower limit will be reached if X is homothetic to B.

Step 1: detection of an anchored homothetic version of B. It is obvious that $X \circ B$ touches X at several border points. These points, called *anchor points*, characterize the opening to some extent. The first step of the algorithm is the detection of anchor points and associated homothetic sets of B. Figure 1 represents X, B and possible anchor points of X respectively. It is not necessary to detect all homothetic sets of B anchored at a point because of a propagation step. For the same reason,

further anchor points will only be detected during the image scan, after the previous propagations.

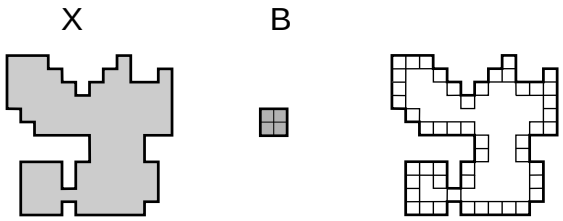

Fig. 1. A set X, the structuring element B and the possible anchor points of X.

Step 2: propagation of B inside X. The propagation process is complex. Eight propagation directions are defined for our purpose which is an opening with a square in 8-connectivity ; the principle is similar for 4 or 6-connectivity openings. The algorithm checks possible propagations in these directions and starts with the larger propagation front. Throughout the propagation in a given direction, new fronts may appear in perpendicular or forward-diagonal directions. They are pushed on a stack until the initial propagation ends. Then each popped front will initiate a new propagation process. When the stack is empty, the algorithm searches for another anchor point.

Example. The first steps of the algorithm are decomposed in Fig. 2. During a horizontal image scan, the algorithm finds the upper left corner of X which is an anchor point and initializes a propagation process. Each propagation direction is checked and the larger front propagates first. In this example, a diagonal propagation is initiated. After propagation step 3, the algorithm detects that the diagonal propagation is no longer possible and changes the direction. But as soon as the front width allows it, the front direction is changed to a diagonal (step 6).

It is easy to adapt the algorithm to the opening of each region of a label image. For this kind of operation, a label propagates only in the corresponding region. All regions will be treated in a single scan in contrast with other methods which would require dealing with each region in turn after binarization.

3.2. Numerical results

The two curves of Fig. 3 compare the linear decomposition with the propagation algorithm for the computation of a binary opening by a square. The given results were obtained on a NeXT 68040 station with `gcc` and `gprof`. The original image was "photograph" of 256x256 pixels, threshold at 128. Note that in the propagation method, the time is independent of the square size. Moreover, also at the origin, the propagation algorithm performs better.

4. Grey-level operations

4.1. Grey-level opening

The key idea of improving the efficiency of an erosion algorithm is that it is useless to recompute a minimum on the set of points representing the structuring element

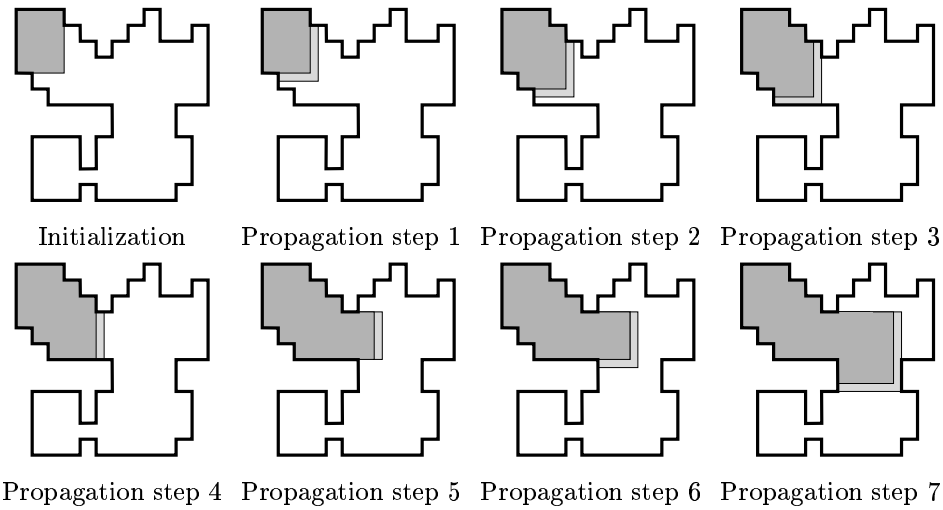

Fig. 2. First steps of a binary opening by propagation. For better understanding, the propagation fronts indicated in light grey are a half pixel wide.

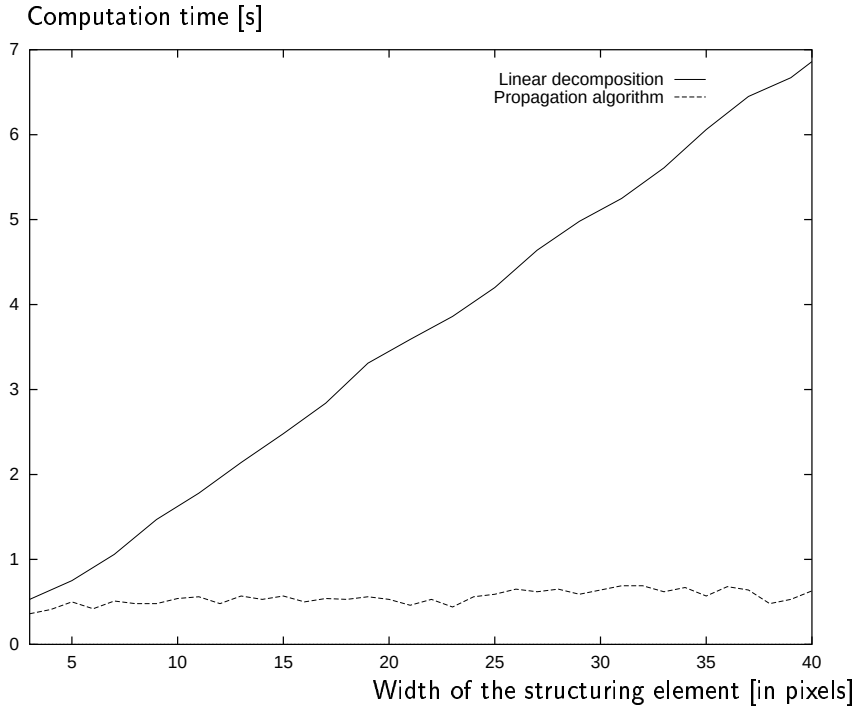

Fig. 3. Computation times of a linear decomposition opening algorithm compared with the proposed propagation algorithm applied on a simple binary image.

for two neighbouring points. This has led HUANG *et al.* [4] to use a sliding window combined with a histogram analysis to compute rank-order filters. For each window position, a pixel of the output image receives the local minimum value. However, if the minimum is compared to each output pixel contained in the window, it is possible to compute an *opening in a single pass.* The general algorithm is the following :

1. All the pixels that enter a window for the first time receive the minimum value.
2. The other pixels receive the maximum between the previously retained value and the local minimum.

This algorithm takes a simplified form when B is a segment because it is easy to track the local minimum values. In that case, the complete step 2 is only needed when the local minimum has increased after the window translation. Otherwise the algorithm attributes the minimum value to the new pixel covered by the window. Fig. 4 illustrates the principle. Other structuring elements lead to similar simplifications.

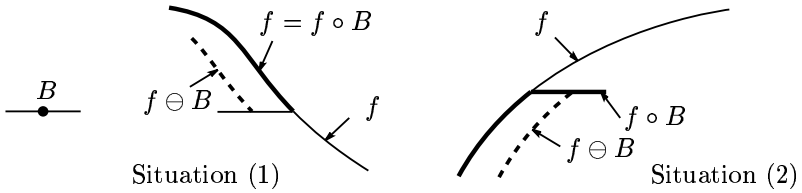

Fig. 4. Two different situations occur during the computation of the opening of f with a linear segment B as the one represented on the left (the origin is located in the middle). Suppose we scan the function from left to right. The negative slope of situation (1) corresponds to a decreasing sliding minimum as is confirmed by the erosion curve $f \ominus B$. It is easy to see that only the opening value of the utmost right point has to be modified after a translation. On the contrary, all values of $f \circ B$ covered by B need to be reallocated in situation (2).

4.2. THEORETICAL RESULTS

In this section, we provide some theoretical results and utilize them to speed up the computation of close-open filters.

An initial theorem establishes a connection between structuring elements involved in grey-level erosions and dilations.

Theorem 1

$$(f \ominus A) \oplus B \leq (f \oplus C) \ominus D \tag{1}$$

if and only if

$$A \oplus C \supseteq B \oplus D.$$

Proof.

The proof is made of two parts:

(1) $\forall f,\ (f \ominus A) \oplus B \leq (f \oplus C) \ominus D \ \Rightarrow\ B \oplus D \subseteq A \oplus C$:

Write o for the origin. $A = o \oplus A \subseteq A$ is equivalent to $o \subseteq A \ominus A$ because $(\ominus, \oplus)$ is an adjunction [3]. If we replace f by A in (1), the relation becomes

$B = o \oplus B \subseteq (A \ominus A) \oplus B \subseteq (A \oplus C) \ominus D$. By the adjunction again, it results in $B \oplus D \subseteq A \oplus C$.

(2) $\underline{\forall f,\ (f \ominus A) \oplus B \leq (f \oplus C) \ominus D \ \Leftarrow\ B \oplus D \subseteq A \oplus C :}$
$\underline{B \oplus D \subseteq A \oplus C}$ implies $(f \ominus A) \oplus B \oplus D \leq (f \ominus A) \oplus A \oplus C$. Moreover, $(f \ominus A) \oplus A \oplus C = (f \circ A) \oplus C \leq f \oplus C$. This means that $(f \ominus A) \oplus B \oplus D \leq f \oplus C$, or $(f \ominus A) \oplus B \leq (f \oplus C) \ominus D$ by the adjunction.

As the proof uses the algebraic framework introduced by HEIJMANS and RONSE [3]), the demonstration scheme is also valid for grey-level erosions and dilations, i.e. with non-flat structuring elements. ■

It is interesting to note that there exists a relationship between structuring elements that induces $(f \ominus A) \oplus B \leq (f \oplus C) \ominus D$. We concentrate below on the combination of openings and closings.

Theorem 2 *If $A \oplus C \supseteq B \oplus C$ and $A \subseteq B$ (which implies that $A \oplus C = B \oplus C$) then*

$$[(f \circ C) \oplus A] \ominus B = (f \circ C) \bullet B.$$

Proof.
If $A \subseteq B$ then $(f \circ C) \oplus A \leq (f \circ C) \oplus B$, and $[(f \circ C) \oplus A] \ominus B \leq (f \circ C) \bullet B$ as erosion is increasing.
On the other hand, $A \oplus C \supseteq B \oplus C \Leftrightarrow C \oplus A \supseteq C \oplus B$, so that we have $(f \oplus A) \ominus B \geq f \circ C$ following relation (1). As this inequality holds for any function f, we may use it for $f \circ C$: $[(f \circ C) \oplus A] \ominus B \geq (f \circ C) \circ C = f \circ C$. It is known that closing is an increasing operation : $([(f \circ C) \oplus A] \ominus B) \bullet B \geq (f \circ C) \bullet B$. Moreover, for any function g, $(g \ominus B) \bullet B = g \ominus B$ (see [2]). In conclusion, $[(f \circ C) \oplus A] \ominus B \geq (f \circ C) \bullet B$. ■

The previous theorem only makes sense if $A \oplus C \supseteq B \oplus C$ and $A \subseteq B$ are realistic. In fact, the induced condition $A \oplus C = B \oplus C$ is not a severe one. Suppose for instance $B = C$. Then the condition becomes $A \oplus B = B \oplus B$, called the *reduction condition*. Fig. 5 shows different sets that satisfy the reduction condition.

In both cases, A is a subset of the border of B. But it is by no means necessary to take all the border points since some of them are redundant. Yet, if A is the border of B, the reduction condition is always satisfied as confirmed by next theorem (the proof can be found in [7]).

Theorem 3 *Let ∂B be the border of the compact set $B \subseteq \mathbf{R}^2$. Then*

$$B \oplus B = B \oplus \partial B$$

If $B = C$, $A \subseteq B$ and $A \oplus C \supseteq B \oplus C$, then $[(f \circ B) \oplus A] \ominus B = (f \circ B) \bullet B$. In other words a dilation by B is replaced by a less expensive dilation by A. Because in alternate sequential filters (a typical case where filtering operators follow one another) the number of involved operations is very high, theorem 2 will prove very useful.

The conditions $A \oplus C \supseteq B \oplus C$ and $A \subseteq B$ do not imply that C is included in B or contains B. Fig. 6 confirms this remark in $\mathbf{Z}^2$.

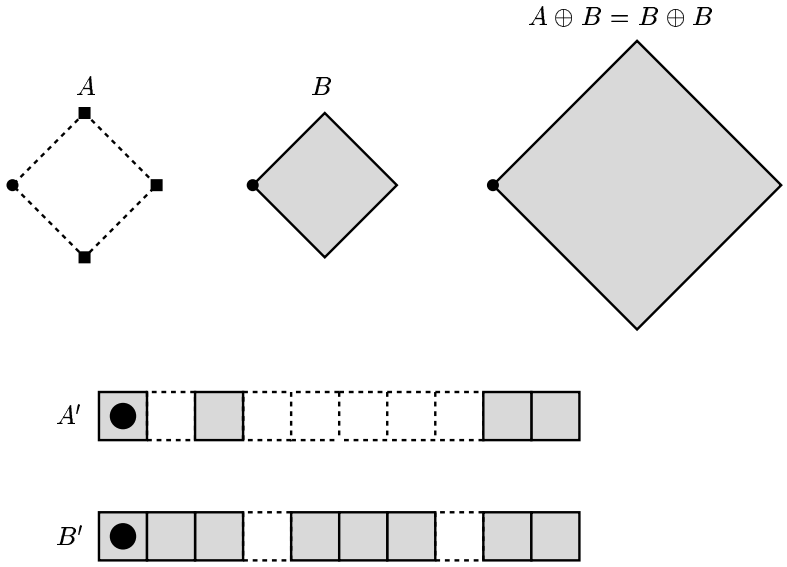

Fig. 5. Two pairs of sets ($A, B \subseteq \mathbf{R}^2$ and $A', B' \subseteq \mathbf{Z}^2$) that satisfy the reduction condition. Each black disk indicates the origin of the corresponding structuring element.

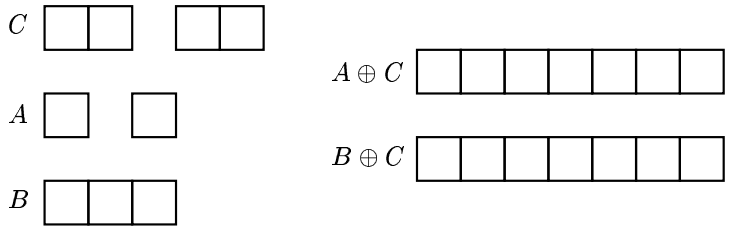

Fig. 6. B and C do not need to be included in each other, even if $A \oplus C \supseteq B \oplus C$ and $A \subseteq B$.

4.3. Numerical results

We have implemented different morphological operations with a horizontal segment. The algorithms are based on a sliding window of which the minimum is the relevant information. All the steps have been integrated so that *only one image scan* is required. The aim of this description is to compare erosion, opening and close-open computation times.

Fig. 7 presents the computation times on a large image (1440x1280). The lower curve corresponds to the erosion algorithm. The second curve represents the time needed for an opening implemented as indicated in section 4.1. With other algorithms, it is twice that of the erosion curve because an opening is an erosion followed by a dilation. Everything happens here as if the dilation was performed during the same scan process as the erosion. The last curve is the close-open filter. Usually this curve is four times the lower one, but due to the combination of the new opening algorithm and theorem 2, it is only about 1.4 higher.

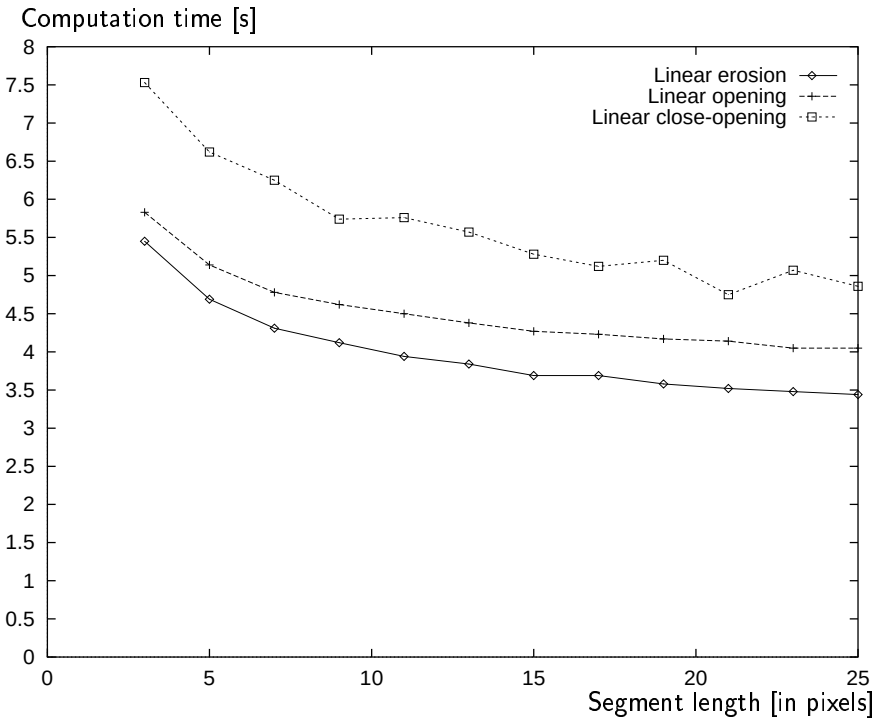

Fig. 7. Computation times of different morphological operations on a large image.

References

1. E. Breen and P. Soille. Generalization of van Herk recursive erosion/dilation algorithm to lines at arbitrary angles. In *Digital Image Computing: Techniques and Applications*, pages 549–555, Sydney, December 1993.
2. R. Haralick, S. Sternberg and X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):532–550, July 1987.
3. H. Heijmans and C. Ronse. The algebraic basis of mathematical morphology; I. Dilations and erosions. *Computer Vision, Graphics, and Image Processing*, 50:245–295, 1990.
4. T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27(1):13–18, February 1979.
5. J. Pecht. Speeding up successive Minkowski operations. *Pattern Recognition Letters*, 3(2):113–117, 1985.
6. M. Schmitt. *Des algorithmes morphologiques à l'intelligence artificielle.* PhD thesis, École nationale supérieure des mines de Paris, February 1989.
7. M. Van Droogenbroeck. *Traitement d'images numériques au moyen d'algorithmes utilisant la morphologie mathématique et la notion d'objet : application au codage.* PhD thesis, Université catholique de Louvain, May 1994.
8. M. van Herk. A fast algorithm for local minimum and maximum filters on rectangular and octogonal kernels. *Pattern Recognition Letters*, 13(7):517–521, July 1992.
9. L. Vincent. Morphological transformations of binary images with arbitrary structuring elements. *Signal Processing*, 22(1):3–23, January 1991.