Fast Algorithms to compute the elementary operators of Mathematical Morphology

JUNIOR BARRERA, ROBERTO HIRATA JR.

Instituto de Matemática e Estatística - USP, Rua do Matão, 1010 - 05389-970 São Paulo - SP - Brasil
email jb@ime.usp.br , hirata@ime.usp.br

Abstract. Mathematical Morphology (**MM**) studies the representation of image operators in terms of some families of simple operators, called the elementary operators of **MM**. A drawback of this approach is that the representation of some operators is not efficient, since it uses a large number of elementary operators. **Fast algorithms** are the solution found by some authors to implement image operators that have complex representations. Two drawbacks of this solution are: i- they change the architecture of the software that implement the operators by the morphological representation. ii- these implementations are very specific and can not be used to implement other operators. Here, we study some of these fast algorithms and show that they can be transformed into morphological representations with equivalent performance, if proper data structures and algorithms are used to implement the elementary operators. Finally, some experimental results that illustrate these ideas are given.

1 Introduction

Mathematical Morphology (**MM**) [5, 6, 3] is a very useful tool for image processing. The central paradigm of **MM** on images is the decomposition of image operators in terms of a formal language, the **Morphological Language** (**ML**), whose vocabulary are the elementary operators **(dilations, erosions)** and operations **(negation, infimum and supremum)** of **MM**. This language is **complete** (i.e. it can represent any function operator) and **expressive** (i.e. many useful operators can be represented as phrases with relatively few words). A phrase of the **ML** is called a **Morphological Operator**. An implementation of the **ML** is called a **Morphological Machine** (MMach) and a program of a MMach is an implementation of a morphological operator on this machine [3]. In this paper, we will be restricted to MMachs implemented on sequential machines.

Any image operator can be implemented in a MMach [1], however, some of these procedures are inefficient, since their representation uses a large number of elementary operators. The solution found by some authors is to implement such procedures in separate by special fast algorithms that are not programs of a MMach. The main drawbacks of this solution is that it changes the architecture of the software that implement the MMach, and that those implementations are very specific and can not be used to implement other operators.

In this paper, we study some of these fast algorithms and show that they can be written as MMach programs with equivalent performance, if proper data structures and algorithms are used to perform the elementary operators and operations.

This new solution does not affect MMach architecture and is more modular, since the fast algorithms for elementary operators and operations can be used in several other morphological operators.

Following this introduction, in section 2, we review some basic concepts of **MM** theory. In section 3, we show some fast algorithms for reconstruction. In section 4, we present equivalent representations for some classical morphological operators, including reconstruction. In section 5, we give new fast algorithms for elementary operators and operations that allow to built MMach programs as efficient as the dedicated algorithms of section 3. In section 6, we present the experimental results. Finally, in section 7, we give some conclusions and future directions for this research.

2 Review of Mathematical Morphology

Let $\mathbb{E}$ be an Abelian group with a binary operation $+$ and an origin o.

Let $K = [0, k]$ be a closed interval in $\mathbb{Z}$ and let $K^{\mathbb{E}}$ be the set of all functions from $\mathbb{E}$ to K. A function $f \in K^{\mathbb{E}}$ represents a binary image if $k = 1$ and a gray-scale image if $k > 1$.

Definition 2.1 *Let $f, g \in K^{\mathbb{E}}$. The operations, for any x in $\mathbb{E}$,*

$$(f \wedge g)(x) = \min\{f(x), g(x)\}$$

and

$$(f \vee g)(x) = \max\{f(x), g(x)\}$$

are called, respectively, **intersection** *and* **union** *of f and g.*

Let X^t be the **transpose** of a subset $X \subset \mathbb{E}$, that is, $X^t = \{x \in \mathbb{E} : -x \in X\}$.

Let $X \subset \mathbb{E}$ and $h \in \mathbb{E}$. We denote by $X + h$, the translation of X by h, i.e., $X + h = \{x \in \mathbb{E}, x - h \in X\}$.

Definition 2.2 *Let $A, B \subset \mathbb{E}$. The operation:*

$$A \oplus B = \cup\{A + b : b \in B\}.$$

is called the **Minkowski addition** *of A and B*

Definition 2.3 *Let $f \in K^{\mathbb{E}}$ and $B \subset \mathbb{E}$. The operators given by, for any x in $\mathbb{E}$,*

$$\delta_B(f)(x) = \max\{f(x + y) : y \in B^t + x\},$$

and

$$\varepsilon_B(f)(x) = \min\{f(x + y) : y \in B + x\}$$

are called, respectively, **dilation** *and* **erosion** *of f by B. The set B is usually called structuring element.*

Definition 2.4 *Let $f \in K^{\mathbb{E}}$, $B \subset \mathbb{E}$ and $n \in \mathbb{Z}^+$. The operators*

$$\delta_B^n(f) = (\delta_B(f))^n = \underbrace{\delta_B(f) \circ \delta_B(f) \circ \ldots \circ \delta_B(f)}_{n \text{ operators}}$$

and

$$\varepsilon_B^n(f) = (\varepsilon_B(f))^n = \underbrace{\varepsilon_B(f) \circ \varepsilon_B(f) \circ \ldots \circ \varepsilon_B(f)}_{n \text{ operators}}$$

are, respectively, the n-**dilation** and the n-**erosion** of f by B.

Definition 2.5 *Let $f \in K^{\mathbb{E}}$, $g \in K^{\mathbb{E}}$ and $B \subset \mathbb{E}$. The operator*

$$\delta_{B,g}(f) = \delta_B(f) \wedge g$$

and

$$\varepsilon_{B,g}(f) = \varepsilon_B(f) \vee g$$

are, respectively, the **dilation** *and the* **erosion** *of f by B* **conditioned** *to g.*

Note that if $f > g$ and $o \in B$, the dilation of f by B conditioned to g will be trivially equal to g. The same happens for the erosion conditioned to g if $f < g$.

The functions f and g are usually called **marker** and **mask** functions, respectively.

Definition 2.6 *Let $f \in K^{\mathbb{E}}$, $g \in K^{\mathbb{E}}$, $B \subset \mathbb{E}$ and $n \in \mathbb{Z}^+$. The operators*

$$\delta_{B,g}^n(f) = (\delta_{B,g})^n(f) = \underbrace{\delta_{B,g}(f) \circ \ldots \circ \delta_{B,g}(f)}_{n \text{ operators}}$$

and

$$\varepsilon_{B,g}^n(f) = (\varepsilon_{B,g})^n(f) = \underbrace{\varepsilon_{B,g}(f) \circ \ldots \circ \varepsilon_{B,g}(f)}_{n \text{ operators}}$$

are, respectively, the n-**dilation** and the n-**erosion** by B **conditioned** to g.

Definition 2.7 *Let $f, g \in K^{\mathbb{E}}$ and $f \leq g$. The iteration until stability of the dilation of f by B conditioned to g, is called the inf-reconstruction operator conditioned to g and it is denoted $\gamma_{B,g}(f)$, that is,*

$$\gamma_{B,g}(f) = \delta_{B,g}^{\infty}(f).$$

Similarly, for $g \leq f$, it is possible to define the sup-reconstruction operator conditioned to g, denoted $\phi_{B,g}(f)$, that is,

$$\phi_{B,g}(f) = \varepsilon_{B,g}^{\infty}(f).$$

Definition 2.8 *The function,*

$$b : \mathbb{E} \to \mathcal{P}(\mathbb{E}),$$

where $\mathcal{P}(\mathbb{E})$ denotes the power-set of $\mathbb{E}$, is called a **structuring function**.

Two particular structuring functions are $b(x) = B^t + x$ and $b(x) = B + x$. The dilation and erosion by a structuring function b, denoted δ_b and ϵ_b, have similar definitions to δ_B and ϵ_B just changing, respectively, $B^t + x$ and $B + x$ by $b(x)$.

Definition 2.9 *Let $B \subset \mathbb{E}$ and $p \in \mathbb{E}$. The function*

$$b_p(x) = \begin{cases} B + x & \text{if } x = p \\ \{x\} & \text{otherwise} \end{cases}$$

is called a **point structuring function**.

A dilation or an erosion by a point structuring function has the property of changing the image just in a point p.

3 Fast Morphological Algorithms

In this section, we will recall some fast algorithms that perform the reconstruction operators defined in last section.

The dimension of the image domain is arbitrary, but the corresponding neighborhood relations must be chosen. The algorithms are written in C pseudo code. The inputs will be a structuring element and an image. The output will be an image, which can be written in the same input image or in a copy of it.

These algorithms use **sequential processing**, **queue processing** or a mixing of them. A **sequential algorithm** has the following properties [4]:

- image pixels are processed in a predefined sequence or order, generally raster or anti-raster;

- the value of the current pixel is written in the same image so it can be used to calculate the value of the next coming pixels.

A queue algorithm has the following properties [7]:

- pixels are stored in a FIFO data structure,
- the order of the pixels in the queue depends on the image,
- the value of the current pixel is written in the same image or in a copy of it.

Some basic operations on a queue W are:

- queue_init(W,n): allocates space for a queue with n points.
- queue_add(W,p): puts the (pointer to) pixel p into the queue W.
- queue_first(W): returns the (pointer to) pixel which is at the beginning of the queue W and removes it.
- queue_empty(W): returns true if the queue W is empty and false otherwise.

A **graph of connectivity** G is a pair $(\mathbb{E}, D)$, where the edges defined by D represent the neighborhood relations between the points of $\mathbb{E}$ [5].

The notation $N_G(p)$ represents the set of all neighbors of a point $p \in \mathbb{E}$ under the graph G[1]. The notations $N_G^+(p)$ and $N_G^-(p)$ represent all the neighbors of the point $p \in \mathbb{E}$ that are accessed, respectively, before and after p in a raster scan. Figures 1 and 2 show these two sets in a grid with connectivity 8.

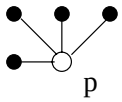

Figure 1: $N_G^+(p)$

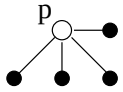

Figure 2: $N_G^-(p)$

3.1 A queue reconstruction algorithm

This algorithm was proposed by Luc Vincent [7] and it is a queue algorithm for binary reconstruction. The idea under this algorithm is to construct a queue of border pixels of the marker and reconstruct the mask image only by border expansions.

In the algorithm bellow, g is the mask image, f is the marker image and $f \leq g$. The result of the reconstruction is written directly in the array representing f.

[1] Note that p is not a neighbor of itself.

- Input: mask image (binary) $g \in [0,1]^{\mathbb{E}}$
  marker image (binary) $f \in [0,1]^{\mathbb{E}}$, $f \leq g$
- Output: result $f \in [0,1]^{\mathbb{E}}$

```
queue_init(W, |E|);
Scan E in raster order
(let p be the current pixel)
   if((f(p) = 1) & (∃q ∈ N_G(p), f(q) = 0)) then
        queue_add(W, p);
while(queue_empty(W) = false)
   p ← queue_first(W);
   for every q ∈ N_G(p)
        if((f(q) = 0) & (g(q) = 1)) then
             f(q) ← 1;
             queue_add(W, q);
return(f);
```

3.2 A hybrid reconstruction algorithm

The algorithm we show bellow was proposed by Luc Vincent [7] and is a hybrid algorithm for binary and gray-scale images. It is the fastest algorithm we have found in the literature for the implementation of gray-scale reconstruction in conventional computers and is called hybrid because it uses aspects of sequential algorithms and of queue based algorithms. The hybrid structure was chosen because both methods alone (pure sequential or pure queue) have some drawbacks.

The idea under this algorithm is simple, it performs a propagation of the marker pixels conditioned to the mask pixels (dilation of the marker conditioned to the mask) by a sequential processing, i.e., it performs first two scannings (raster and anti-raster) and during the last scanning (anti-raster) a queue is built. This queue will hold every pixel whose current value can still be propagated. The other part of the algorithm is a breadth first propagation of the built queue.

In the algorithm bellow, g is the mask image, f is the marker image (defined on the same domain of g) and $f \leq g$. The result of the reconstruction is written directly in the array representing f.

- Input: mask image $g \in K^{\mathbb{E}}$
  marker image $f \in K^{\mathbb{E}}$, $f \leq g$
- Output : result $f \in K^{\mathbb{E}}$

```
Scan E in raster order
(let p be the current pixel)
   f(p) ← max{f(x) : x ∈ N_G^+(p) ∪ {p}} ∧ g(p)};
Scan E in anti-raster order
(let p be the current pixel)
   f(p) ← max{f(x) : x ∈ N_G^-(p) ∪ {p}} ∧ g(p)};
   if(∃q ∈ N_G^-(p), (f(q) < f(p)) & (f(q) < g(q)))
   then queue_add(W, p);
while(queue_empty(W) = false)
   p ← queue_first(W);
   for every q ∈ N_G(p)
        if((f(q) < f(p)) & (f(q) ≠ g(q))) then
             f(q) ← min{f(p), g(q)};
             queue_add(W, q);
return(f);
```

4 Equivalent Representations

Inspired in the algorithms presented in the last section, we propose here new equivalent definitions for the elementary operators of **MM** and show how they can be used for the representation of the inf-reconstruction.

Let first give some basic definitions. Let $\mathbb{B}$ denote the subset $N_G(o) + \{o\}$.

Definition 4.1 *Let $f \in K^{\mathbb{E}}$ and $B \subset \mathbb{B}$. The frontier of f relatively to B is the subset ∂f, given by,*

$$\partial f = \{x \in \mathbb{E} : \exists p \in B + x, f(p) < f(x)\}$$

4.1 Representation of queue algorithms

The algorithms based on queues can be described formally by new representations that apply the usual elementary operators of **MM** only to the points of the frontier of the images.

4.1.1 Dilation

To compute the dilation as defined in section 2, we have to compute the value associated to all points of $\mathbb{E}$. However, this computation usually will change the value of just a few points. This is better seen in figure 3, where it is shown a gray-scale image and its dilation by a structuring element $\mathbb{B}$, considering a grid with connectivity 4.

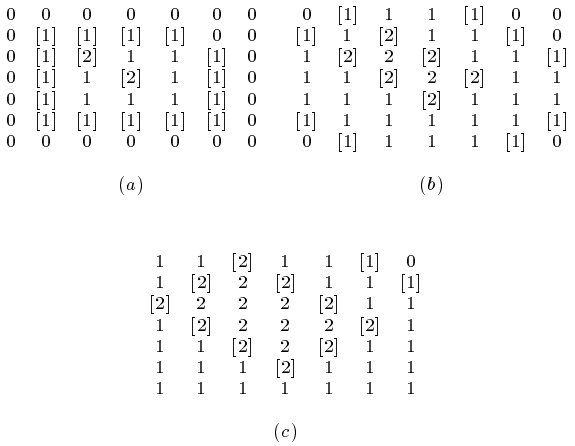

Figure 3: Gray-scale image (a), its first (b) and second (c) dilation by the structuring element $\mathbb{B}$ on a square grid with connectivity 4

The points that have been changed by the dilation are the neighbors of the points marked with brackets (i.e., the frontier points). Therefore, a new equivalent definition for dilation could depend only on the frontier points ∂f and their neighbors.

Definition 4.2 *Let $f \in K^{\mathbb{E}}$ and $B \subset \mathbb{B}$. The dilation of f by B is given by, for all $x \in \mathbb{E}$,*

$$\delta_B(f)(x) = \begin{cases} \max\{f(y), y \in B^t + x \cap \partial f\}, \\ \qquad\qquad if\ x \in \partial f \oplus B \\ f(x) \quad otherwise \end{cases}$$

Note that to compute a second dilation, we can easily compute a new frontier $\partial f'$, where $f' = \delta_B(f)$, from ∂f and f'. It is enough to take the points $x \in B + p$, $p \in \partial f$, such that $\exists q \in B + x$ where $f'(q) < f'(x)$ (see figures 3 b and 3 c). In other words: $\partial f' = \{x \in \partial f \oplus B : \exists q \in B + x, f'(q) < f'(x)\}$.

This idea can be used recursively to compute the n-dilation of a function f by a structuring element B:

$$\begin{aligned} f_{i+1} &= \delta_B(f_i) \\ \partial f_{i+1} &= \{x \in \partial f_i \oplus B : \exists p \in B + x, \\ &\qquad f_{i+1}(p) < f_{i+1}(x)\} \end{aligned}$$

where, $f_0 = f$, $\partial f_0 = \partial f$, and $\delta_B^n(f) = f_n$.

A useful particularization of the infimum operation is given by the following definition.

Definition 4.3 *Let $f, g \in K^{\mathbb{E}}$ such that $f \leq g$. The infimum of the functions $h = \delta_B(f)$ and g is given by, for any $x \in \mathbb{E}$,*

$$(h \wedge g)(x) = \begin{cases} min\{h(x), g(x)\}, & if\ x \in \partial f \oplus B \\ h(x) & otherwise \end{cases}$$

Of course, the conditional dilation (and, consequently, the inf-reconstruction) can be built from the definitions above.

Similar ideas can be used to create a new representation for erosion, n-erosion, supremum, conditional erosion and sup-reconstruction.

4.2 Representation of Sequential Algorithms

As we have seen in section 3, sequential algorithms [4] change the value of just one point at a time and the computation of this value can use the values of the points computed before it. This process can be described by operators defined by point structuring functions. We will show how they can be composed to perform the raster or anti-raster pixel processing of the hybrid algorithms. Finally, we will show how to write the sequential reconstruction as a morphological operator.

Definition 4.4 *Let B^+ denote the subset $N_G^+(o) \cup \{o\}$. The raster structuring function for the point $p \in \mathbb{E}$ is given by, for any $x \in \mathbb{E}$,*

$$b_p^+(x) = \begin{cases} B^+ + x & \text{if } x = p \\ \{x\} & \text{otherwise} \end{cases}.$$

Definition 4.5 *Let B^- denote the subset $N_G^-(o) \cup \{o\}$. The anti-raster structuring function for the point $p \in \mathbb{E}$ is given by, for any $x \in \mathbb{E}$,*

$$b_p^-(x) = \begin{cases} B^- + x & \text{if } x = p \\ \{x\} & \text{otherwise} \end{cases}.$$

Definition 4.6 *Let $f, g \in K^{\mathbb{E}}, f \leq g$ and $B \subset \mathbb{B}$. The operator $\Psi_{B,g}^+$ on $K^{\mathbb{E}}$, called raster reconstruction, is given by $\Psi_{B,g}^+(f) = f_{p_{|E|}}$,*

$$f_{p_{i+1}} = \delta_{b_{p_i}^+}(f_{p_i}) \wedge g, \text{ where}$$

$p_1, p_2, \ldots, p_{|E|}$ is the sequence of points of $\mathbb{E}$ in the raster order and $f_{p_1} = f$.

The raster reconstruction operator is a phrase of the **ML** that represents the raster processing in the fast reconstruction algorithm.

Analogously, changing the structuring function b_p^+ by the structuring function b_p^- and taking the points of $\mathbb{E}$ in the anti-raster order, we define the anti-raster reconstruction operator $\Psi_{B,g}^-$ that represents the anti-raster processing in the fast reconstruction algorithm.

Next, we will present a useful specialization of the infimum for sequential algorithms.

Definition 4.7 *Let $f, g \in K^{\mathbb{E}}$ such that $f \leq g$, and $p \in \mathbb{E}$. The infimum of $h = \delta_{b_p}(f)$ and g is given by, for any $x \in \mathbb{E}$,*

$$(h \wedge g)(x) = \begin{cases} \min\{h(x), g(x)\} & \text{if } x = p \\ h(x) & \text{otherwise} \end{cases}$$

The definition for supremum is similar.

Definition 4.8 *The sequential reconstruction operator is the operator given by the following composition,*

$$\Psi_{B,g} = \Psi_{B,g}^- \circ \Psi_{B,g}^+$$

The operator $\Psi_{B,g}$ represents the sequence of a raster and an anti-raster reconstruction algorithm.

The sequential reconstruction does a series of raster, anti-raster processing until stability. Therefore, applying the operator $\Psi_{B,g}$ until stability we get the inf-reconstruction of the function f conditioned to g.

4.3 Representation of Hybrid Algorithms

It is experimentally known that a hybrid reconstruction (merge of the sequential reconstruction and of the queue based reconstruction) is more efficient than any of them alone. The equivalent representation of this operator is formed by the composition of the sequential reconstruction operator with a queue based reconstruction operator.

Definition 4.9 *Let $f, g \in K^{\mathbb{E}}$, $f \leq g$ and $n \in \mathbb{Z}^2$. The hybrid reconstruction of g from f is given by the following composition,*

$$\rho_{B,g}(\Psi_{B,g}^n(f)),$$

where $\rho_{B,g}$ is the queue based reconstruction and $\Psi_{B,g}$ is the sequential reconstruction.

Note that in the queue reconstruction we need to take just the points of $\Psi_{B,g}(f)$ such that,

$$\Psi_{B^+,g}(f)(p) > \Psi_{B^+,g}(f)(q) \quad \text{and} \quad \Psi_{B^+,g}(f)(q) < g(q),$$

It is easy to see that the inclusion of the other points of $\partial\Psi_{B,g}$ would not change the final result.

5 Algorithms for the Equivalent Representations

The representations described in the previous section will be used to built fast algorithms. They are presented in C pseudo code.

In general, f will represent an input image, g an input mask image, B a structuring element, ∂f and $\partial f'$ queues to keep the border of, respectively, a function f and the result of its transformation f'.

5.1 Queue based algorithms

In this class of algorithms, the images are represented by an array of pixels and a queue of interesting pixels.
To compute the subset of the border that initializes the hybrid reconstruction, we need the following algorithm.

```
Procedure hybborder(f, g, B, ∂f)
Scan 𝔼 in anti-raster order
(let p be the current pixel)
  if (∃ q ∈ B + p, f(q) < f(p) and f(q) < g(q))
    then queue_add(p);
```

The algorithm to find the complete border is similar, just eliminating the second inequality.

5.1.1 Infimum

By definition 4.3 the infimum between h and g, where $h = \delta_B(f)$ and $f \leq g$ can be performed by the following algorithm.

```
Procedure infqueue(h, g, B, ∂f)
while (queue_empty(∂f) = false);
   p ← queue_first(∂f);
   for every q ∈ B + p
      if (h(q) > g(q)) then h(q) ← g(q);
```

Note that the border of $h \wedge g$ is a subset of the border of h, since $h \wedge g \leq h$. Hence, it can be used in a following process.

The algorithm for the supremum is quite similar to the one above.

5.1.2 Dilation

Using the definition 4.2, we can easily write an algorithm for dilation of a function f by a structuring element B.

```
Procedure dilqueue(f, B, ∂f, ∂f')
while (queue_empty(∂f) = false)
    p ← queue_first(∂f);
    for every q ∈ B + p
        if (f(q) < f(p)) then f(q) ← f(p);
        if (∃r ∈ B + q, f(r) < f(q))
          then queue_add(∂f', q);
```

5.1.3 Queue Reconstruction

This algorithm for reconstruction is similar to its representation given by definition 2.7.

```
Procedure queuerec(f, g, B, ∂f)
while (queue_empty(∂f) = false)
      dilqueue(f, B, ∂f, ∂f');
      infqueue(f, g, B, ∂f');
      ∂f ← ∂f';
      ∂f' ← null;
```

5.2 Algorithms based on sequential processes

In this class of algorithms, images are represented by an array of pixels and a particular pixel. The implementation of most of these algorithms are very easy and their complexity are constant. The sequential algorithms for infimum, supremum, erosion and dilation are straightforward so we will not show them here. The sequential infimum and dilation have the following prototypes: inf_seq(f, g, p) and dil_seq(f, B, p).

5.2.1 Sequential Reconstruction

This algorithm will implement the operator defined in section 4.2. It is very similar to the first part of the algorithm described in section 3.2. A sequence of scannings is done and in the final of each scanning it is verified if the image was modified or not. The algorithm finishes when the image was not modified after a scanning.

```
Procedure rec_seq(f, g, B)
repeat until stability
   Scan 𝔼 in raster order
   (let p be the current pixel)
       dil_seq(f, B+, p);
       inf_seq(f, g, p);
   Scan 𝔼 in anti-raster order
       (let p be the current pixel)
       dil_seq(f, B−, p);
       inf_seq(f, g, p);
```

5.3 Hybrid Algorithms

Here we will show an equivalent algorithm for the fast hybrid reconstruction presented in section 3.

5.3.1 Hybrid Reconstruction

This algorithm implement the operator described in section 4.3.

```
Procedure rec_hyb(f, g, B)
Scan 𝔼 in raster order
    (let p be the current pixel)
    dil_seq(f, B+, p);
    inf_seq(f, g, p);
Scan 𝔼 in anti-raster order
(let p be the current pixel)
    dil_seq(f, B−, p);
    inf_seq(f, g, p);
hybborder(f, g, B−, ∂f)
while (queue_empty(∂f) = false)
    dilqueue(f, B, ∂f, ∂f');
    infqueue(f, g, B, ∂f);
    ∂f ← ∂f';
    ∂f' ← null;
```

6 Experimental Results

In this section we will show some comparative measures of the efficiency of these algorithms.

The algorithms presented in the last section have been implemented in **ANSI C** and integrated to **Khoros** to be tested. The tests have been made in a PC Pentium 90MHz with 40Mb of RAM memory.

Each experiment has been repeated 10 times and the graphics show the mean time of the measures. All the input images are over 250.000 points. The image used in the experiments with binary images is composed of 108 small discs and ellipses. The image used in the experiments with gray-scale images is shown in fig. 4.

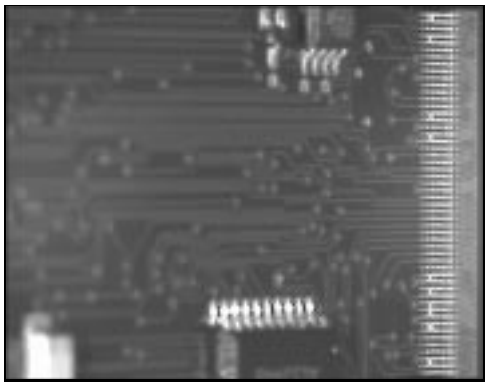

Figure 4: Gray-scale image

6.1 Results for Dilation

In each experiment of this section we have applied from 1 to 100 dilations in the input image. We have compared the following algorithms:

- MMACH-BIT - this is the dilation algorithm implemented in MMach to process binary images in bit

compacted format (i.e. each point of the image are represented by 1 bit) [2].

- MMACH - this is the dilation algorithm in MMach for images whose formats are byte or short [2].

- QUEUE - this is the dilation algorithm implemented by queues.

6.1.1 Binary Dilation

In this experiment we have used a 512×512 binary image as input and we varied the number of dilations. The graphic on fig. 5 shows the result of one of these experiments for a 3×3 structuring element.

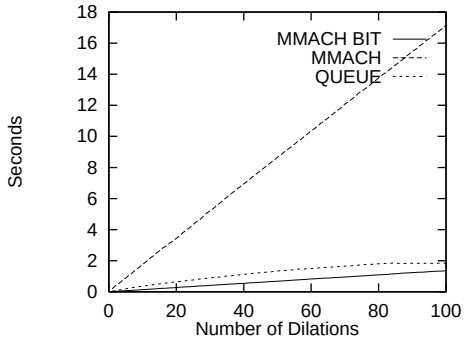

Figure 5: Total time for binary dilations

The MMACH-BIT algorithm is highly optimized and it is machine dependent, so it is very difficult to beat it. The QUEUE algorithm has a very good performance considering that it is easily implemented and is not machine dependent. The MMACH algorithm is optimized for grayscale images but its performance is not good for binary images.

6.1.2 Grayscale Dilation

In this experiment we have used a 511×400 grayscale image as input (see fig. 4) and we varied the number of dilations. The graphic of fig. 6 shows the result of one of these experiments for the 3×3 cross (i.e., the cross included in the 3×3 structuring element.)

The MMACH algorithm has a slightly better performance than the QUEUE one until the 18 th dilation. Then its performance is very better than the MMACH mainly for a large number of dilations.

6.2 Results for Conditional Dilation

In the experiments of this section we have applied from 1 to 50 conditional dilations in the input image. We have compared the following algorithms:

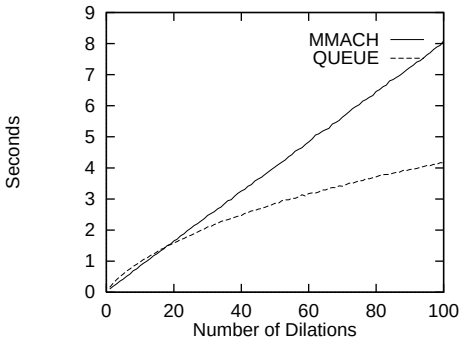

Figure 6: Total time for grayscale dilations

- MMACH - this is the conditional dilation algorithm in MMach for images whose formats are byte or short [2].

- QUEUE - this is the conditional dilation algorithm implemented by queues. It is modular like MMACH algorithm, i.e., it performs a dilations and then an infimum in the input images.

- BETTER QUEUE - this is a non modular conditional dilation algorithm implemented by queues, i.e., it performs a dilation followed by an infimum for each point of the image.

We will show only one of our results for grayscale conditional dilation.

6.2.1 Grayscale Conditional Dilation

In this experiment we have used a N-erosion of the mask image as the marker image. We have performed N-conditional dilations on this image. The graphic on fig. 7 shows the result of one of these experiments for the 3×3 cross structuring element.

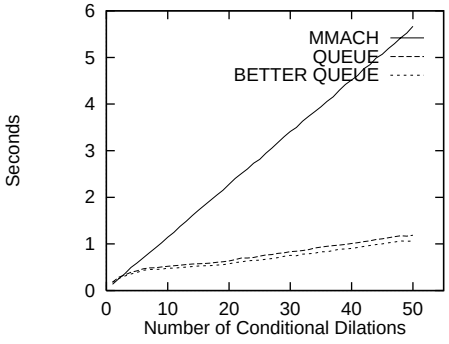

Figure 7: Total time for conditional dilation

The MMACH algorithm has a slightly better performance for one dilation but QUEUE's performance is for more than one dilation.

6.3 Results for Reconstruction

This experiment is similar to the one of conditional dilation, but it is necessary to compare the performance of our algorithms to the fast reconstruction algorithms which motivated them. We have compared four algorithms:

- MMACH-CLAS - this is an algorithm implemented according to the original definition, i.e., iteration of conditional dilations until stability.
- MMACH - this is the fast reconstruction algorithm (section 3.2) implemented in MMach for images whose formats are byte or short [2].
- QUEUE - this is the algorithm of section 5.3.1 implemented according to the definition, but using conditional dilation implemented by queues.
- BQUEUE - this algorithm is similar to QUEUE, but using the BETTER QUEUE conditional dilations.

6.3.1 Grayscale Reconstruction

The graphic on fig. 8 shows the result of the inf-reconstruction using as structuring element the 3×3 cross.

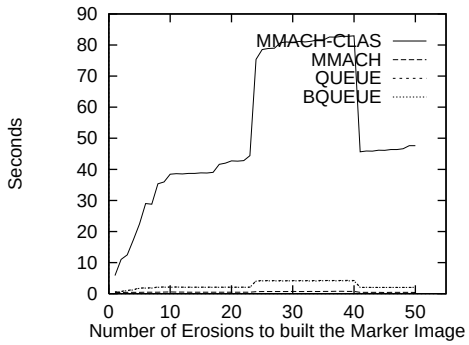

Figure 8: Total time for reconstruction

The MMACH algorithm has the best performance of all, as expected. The new morphological operators (their curves are overwritten) have a similar but worse performance. However, their performance is much better than the MMACH-CLAS algorithm.

7 Conclusions

We have studied fast algorithms for image processing and shown how to represent them as morphological operators. For this representation, we have introduced some new elementary operators and given adequate data structures and algorithms for their implementation.

Experimental results have shown that the proposed morphological operators have performance equivalent to the fast algorithms studied. We should remark that both types of algorithms depend on the input image. They perform better for images that have smaller amount of borders. As erosions and dilations diminishes the number of borders, these algorithms are particularly adequate to high resolution images that impose the use of large sequences of erosions and dilations.

This approach conserves the hierarchical architecture of the MMachs and permits the efficient representation of other morphological operators, since they are built by compositions of elementary operators.

The idea of implementing fast algorithms as morphological operators could be applied to other known fast algorithms, increasing their modularity and therefore increasing their utility.

8 Acknowledgments

This project was supported by ProTeM-CC/CNPq through the AnIMoMat project, contract 680067/94-9. The authors also have a partial support from Olivetti do Brasil.

References

[1] G. J. F. Banon and J. Barrera. Decomposition of mappings between complete lattices by mathematical morphology: Part I. general lattices. *Signal Processing*, 30:299–327, 1993.

[2] J. Barrera, G. F. Banon, R. A. Lotufo, and R. Hirata Jr. MMach: a Mathematical Morphology Toolbox for the KHOROS System. *Submited to Jornal of Eletronic Image*, 1997.

[3] J. Barrera, F. S. C da Silva, and G. J. F. Banon. Automatic programming of binary morphological machines. In *Image Algebra and Morphological Image Processing V*, volume 2300, pages 229–240. SPIE, San Diego, 1994.

[4] A. Rosenfeld and J. L. Pfaltz. Sequential Operations in Picture Processing. *Journal of the Association for Computing Machinery*, pages 471–494, 1966.

[5] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

[6] J. Serra, editor. *Image Analysis and Mathematical Morphology. II: Theoretical Advances*. Academic Press, London, 1988.

[7] L. Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Trans. on Image Processing*, 2(2):176–201, April 1993.