Jump-Miss Binary Erosion Algorithm

Anderson Fraiha Machado and Ronaldo Fumio Hashimoto

Institute of Mathematics and Statistics of University of Sao Paulo, SP, Brazil
dandy@ime.usp.br and ronaldo@ime.usp.br

Abstract

This work presents a new and fast algorithm for binary morphological erosions with arbitrary shaped structuring elements inspired by preprocessing techniques that are quite similar to those presented in many fast string matching algorithms (jumps and miss-matchings). The result of these preprocessing techniques is a speed up for computing binary erosions. A time complexity analysis shows that this algorithm has clear advantages over some known implementations. Experimental results confirm this analysis and shows that this algorithm has a good performance and can be a better option for erosions computation.

I. Introduction

Mathematical Morphology (MM) is a general method for studying mappings between complete lattices. In particular, mappings between binary images are of special interest and they are called *set operators* [1]. A central paradigm in MM is the representation of set operators in terms of dilations, erosions, unions, intersections, complementations and compositions. For example, it is well-known that increasing set operators can be represented as a union of erosions. Since erosion is the morphological dual to dilation, a fast erosion algorithm can also produce an efficient implementation of dilations. In many applications (for example, granulometries), the time efficiency of set operators is quite important. In this paper, we present an efficient algorithm for binary erosion with arbitrary shaped structuring elements inspired by preprocessing techniques which is quite similar to those presented in many fast string matching algorithms (jumps and miss-matchings). A time complexity analysis shows that this algorithm has clear advantages over some known erosion implementations. Experimental results confirm this analysis and show that this algorithm has a good performance and it can be a better option for erosion (and dilation) computation.

II. Binary Mathematical Morphology

Let $\mathbb{Z}$ denote the integer set. Let $P(\mathbb{Z}^d)$ denote the power set of $\mathbb{Z}^d$. Let $X, Y \subseteq \mathbb{Z}^d$. The operations $X \cup Y$, $X \cap Y$, $X \backslash Y$ and X^c are the usual set operations of union, intersection, difference and complementation, respectively. Let $\subseteq$ denote the usual set inclusion relation. The pair $(P(\mathbb{Z}^d), \subseteq)$ is a *complete Boolean lattice* [2].

For the purpose of computational time analysis, we need to define the data structures that are used to store the subsets of $\mathbb{Z}^d$. Although there many ways to store sets, in this paper, a set $X \subseteq \mathbb{Z}^d$ is stored using either a *linked list* (denoted by L_X), or a *bound multidimensional array* (denoted by M_X), which is a multidimensional matrix representing some portion of the $\mathbb{Z}^d$ grid that contains X. When using linked list, points in X are directly stored in L_X (that is, only points $x \in X$ are stored in L_X) and, consequently, $|L_X| = |X|$. When using bound multidimensional array M_X, the points are stored in the following way:

$$M_X(z) = \begin{cases} 1 & z \in X, \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, $|M_X|$ is the size of the portion grid of $\mathbb{Z}^d$ that contains X and it may be larger than $|X|$.

Let $(\mathbb{Z}^d, +)$ be an *Abelian group* with zero element $o \in \mathbb{Z}^d$, called *origin*. Let $h \in \mathbb{Z}^d$ and $X \subseteq \mathbb{Z}^d$. The set $X_h = \{x + h : x \in X\}$ is the *translation* of X by h. The set $X^t = \{-x : x \in X\}$ is the *transpose* of X. We say that X is *symmetric* if and only if $X = X^t$.

Let $X, B \subseteq \mathbb{Z}^d$. The set operations $X \oplus B = \cup_{b \in B} X_b$ and $X \ominus B = \cap_{b \in B} X_{-b}$ are called *Minkowski addition* and *Minkowski subtraction*, respectively. The set operations $X \circ B = (X \ominus B) \oplus B$ and $X \bullet B = (X \oplus B) \ominus B$ are called *opening* and *closing*, respectively. The operators $\delta_B : P(\mathbb{Z}^d) \to P(\mathbb{Z}^d)$ and $\varepsilon_B : P(\mathbb{Z}^d) \to P(\mathbb{Z}^d)$ defined by $\delta_B(X) = X \oplus B$ and $\varepsilon_B(X) = X \ominus B$, $\forall X \subseteq \mathbb{Z}^d$, are, respectively, the *dilation* and *erosion* by the *structuring element* (SE) B. A well-known property for erosions and

dilations is that if $o \in B$, then $\varepsilon_B(X) \subseteq X \subseteq \delta_B(X)$. Another property is that if $b \in B$, then $\varepsilon_B(X) = \varepsilon_{B_{-b}}(X_{-b})$. Thus, we can compute $\varepsilon_B(X)$ by computing $\varepsilon_{B_{-b}}(X_{-b})$, where b is any point of B. Note that $o \in B_{-b}$. Then, in order to compute $\varepsilon_{B_{-b}}(X_{-b}) \subseteq X_{-b}$, it is sufficient to find the points $h \in X_{-b}$ in which $B \subseteq X_{-b-h}$. Thus, for the sake of simplicity, in this paper, we consider that $o \in B$. Dilations and erosions are closely related. They are dual operators of each other, that is, $\varepsilon_B(X) = (\delta_B(X^c))^c$ and $\delta_B(X) = (\varepsilon_B(X^c))^c$ [1], [3], [4]. Therefore, as they are dual operators of each other, an efficient implementation of one of them may produce an efficient implementation of the other. In this paper, we choose to investigate an efficient implementation for the erosion operator. For that, we use the equivalent definition [1], that is,

$$\varepsilon_B(X) = \{h \in \mathbb{Z}^d : B_h \subseteq X\}. \quad (1)$$

In the following subsections, we present two well-known important binary morphological operators (erosion transform and morphological skeleton) which will be used in the preprocessing step for our erosion algorithm.

A. Erosion Transform

The erosion transform of a set X by a SE A, is a grayscale image $f_X^A : \mathbb{Z}^d \to \mathbb{Z}$ built from a successive morphological erosions. It is a generalization of the distance transform [5]. Let us formally define the erosion transform.

Definition 2.1 (n-fold Minkowski subtraction): Let $X, A \subseteq \mathbb{Z}^d$ and $n \in \mathbb{Z}$. The *n-fold Minkowski subtraction* of X by A, denoted by $(X \ominus_n A)$, is defined as

$$X \ominus_n A = \begin{cases} \emptyset & \text{if } n < 0, \\ X & \text{if } n = 0, \\ X \underbrace{\ominus A \ominus A \ominus \cdots \ominus A}_{n \text{ operations}} & \text{if } n > 0. \end{cases} \quad (2)$$

The *n-fold erosion* of a set X by a SE A is defined as $\varepsilon_A^n(X) = X \ominus_n A$.

Definition 2.2 (Erosion Transform): The *erosion transform* of X by A is the grayscale image $f_X^A : \mathbb{Z}^d \to \mathbb{Z}$ defined as

$$f_X^A(z) = \begin{cases} \max\{n : z \in (X \ominus_{n-1} A)\} & \text{if } z \in X, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

It is very known that once f_X^A has been computed, it only requires a simple thresholding to compute the binary erosion $\varepsilon_A^n(X)$ [6]. In fact, the idea to use a sort of distance transform and to threshold it in order to compute erosions was proposed in 1987 by Lay [7]. However, it is important to keep in mind that it is not the case here. We are just computing f_X^A as preprocessing step in order to compute $\varepsilon_B(X)$ for a given SE B (with arbitrary shape). Typically, in this paper, $A \neq B$ and is a subset of the 3×3 window centered at the origin. Although, A can be chosen according to problem situation, the only requirement is that $|A|$ must be a small and constant. In order to avoid any confusion between A and B, we will call A as *adjacent set*. In 1995, Chen and Haralick [6] presented a fast recursive algorithm to compute f_X^A. In this section, we present an algorithm (very similar to Chen and Haralick's algorithm) to compute the erosion transform (see Algorithms 1 and 2).

Algorithm 1 Erosion_Transform (X, A)

Input: a set X stored as M_X and an adjacent set A stored as L_A.
Output: the erosion transform f_X^A.

```
1: for all z ∈ M_X do
2:     f_X^A(z) ← +∞;
3: end for
4: for all v ∈ L_A do
5:     for all z ∈ M_X do
6:         f_X^A(z) ← Get_Value (X, v, z);
7:     end for
8: end for
```

Algorithm 2 Get_Value (X, v, z)

Input: a set X stored as M_X, a point $v \in \mathbb{Z}^2$ and a point $z \in X$.
Output: the value of $f_X^A(z)$.

```
1: if M_X(z) = 0 /* that is, z ∉ X */ then
2:     return 0;
3: end if
4: return min{f_X^A(z), Get_Value (X, v, z + v) + 1};
```

B. Time Complexity Analysis of Erosion Transform Algorithm

In order to compute $f_X^A(z)$, the algorithm **Get_Value** (X, v, z) runs in $O(|M_X| \cdot |L_A|)$ and so does the algorithm **Erosion_Transform**. Observe that once a point p has been visited by **Get_Value**, one can obtain $f_X^A(p)$ in $\Theta(1)$. If $|A|$ is a constant, the algorithm **Erosion_Transform** runs in linear time, that is, it runs in $\Theta(|M_X|)$.

C. Morphological Skeleton

The morphological skeleton of an image is useful for binary shape recognition and compression [8]. Here, we will use it as a preprocessing step for erosion computation. Let us formally define the morphological skeleton of an image.

Definition 2.3 (n-fold Minkowski addition): Let $X, A \subseteq \mathbb{Z}^d$ and $n \in \mathbb{Z}$. The n-fold Minkowski addition of X by A, denoted by $(X \oplus_n A)$, is defined as

$$X \oplus_n A = \begin{cases} \emptyset & \text{if } n < 0, \\ X & \text{if } n = 0, \\ X \underbrace{\oplus A \oplus A \oplus \cdots \oplus A}_{n \text{ operations}} & \text{if } n > 0. \end{cases} \quad (4)$$

A *disk* of radius n centered at $z \in \mathbb{Z}^d$ using an adjacent set A is defined as $\{z\} \oplus_n A$. Let $X, A \subseteq \mathbb{Z}^d$ and $z \in \mathbb{Z}^d$. We define $D_X^A(z) \subseteq X$ as the set

$$D_X^A(z) = \begin{cases} \emptyset, & \text{if } z \notin X; \\ (\{z\} \oplus_m A), & \text{otherwise,} \end{cases}$$

where $m = \max\{n : (\{z\} \oplus_n A) \subseteq X\}$. Note that, if $z \in X$, $D_X^A(z)$ is the *maximal disk centered at* z using the adjacent set A within X. Now, let us define maximal disks within X. Given a point $x \in X$, if there is no other disk besides $D_X^A(x)$ lying within X and properly containing $D_X^A(x)$, then we say that $D_X^A(x)$ is a *maximal disk*. The centers of all maximal disks comprise the *morphological skeleton* of the image. The morphological skeleton of an image X using an adjacent set A, denoted by S_X^A, can be obtained by using Lantuejoul's formula [1], [9]:

$$S_X^A = \bigcup_{m=0}^{\infty} (X \ominus_m A) \setminus [(X \ominus_m A) \circ A]. \quad (5)$$

A direct implementation of this equation yields an algorithm with high computational time complexity. Maragos and Schafer [8] presented a fast (linear) algorithm for S_X^A computation, but it works only when A is a convex symmetric adjacent set. In this section, we present an efficient algorithm to compute S_X^A using the erosion transform f_X^A without imposing any restriction on the adjacent set A. The next proposition gives some properties of skeleton and erosion transform [1], [8].

Proposition 2.4: Let $X, A \subseteq \mathbb{Z}^d$. Then following statements hold: (i) $S_X^A \subseteq X$, if $o \in A$; (ii) $D_X^A(p) \subseteq X$, $\forall p \in \mathbb{Z}^d$; and (iii) $X = \bigcup_{s \in S_X^A} D_X^A(s)$.

The last statement of Proposition 2.4 corresponds to the reconstruction from morphological skeleton [8]. As a direct consequence of the last statement of Proposition 2.4, we have the following proposition.

Proposition 2.5: Let $X, A \subseteq \mathbb{Z}^d$. Then, for each $p \in X$, there exists $s \in S_X^A$ such that $p \in D_X^A(s)$.

Let $X, A \subseteq \mathbb{Z}^d$. Given a point $z \in X$, the next proposition provides a relationship between $D_X^A(z)$ and the erosion transform f_X^A.

Proposition 2.6: Let $X, A \subseteq \mathbb{Z}^d$. If $z \in \mathbb{Z}^d$, then $D_X^A(z) = \{z\} \oplus_{m-1} A$, where $m = f_X^A(z)$.

Algorithm 3 Skeleton (X, A)

Input: an image X stored as both M_X and L_X, and an adjacent set A stored as L_A.

Output: the morphological skeleton S_X^A and the erosion transform f_X^A.

```
1: S_X^A ← ∅;
2: f_X^A ← Erosion_Transform (X, A);
3: for all x ∈ L_X do
4:    if max{f_X^A(x + v) : v ∈ L_{A^t}} ≤ f_X^A(x) then
5:       S_X^A ← S_X^A ∪ {x};
6:    end if
7: end for
8: return (S_X^A, f_X^A);
```

Proof: By definition of erosion transform, if $z \notin X$, then $f_X^A(z) = 0$ and, consequently, $D_X^A(z) = \{z\} \oplus_{m-1} A = \emptyset$, since $m = 0$. So, assume that $z \in X$. In this case, we need to show that these two statements hold: (i) $(\{z\} \oplus_{m-1} A) \subseteq X$ and (ii) $(\{z\} \oplus_m A) \not\subseteq X$. Now, let us prove (i). By definition of erosion transform, $\{z\} \subseteq (X \ominus_{m-1} A)$. Since dilation (and consequently, Minkowski addition) is an increasing operator [10], $(\{z\} \oplus_{m-1} A) \subseteq (X \ominus_{m-1} A) \oplus_{m-1} A$. Note that the right side of the previous inclusion is an opening operator. Since opening is an anti-extensive operator [10], we have that $(\{z\} \oplus_{m-1} A) \subseteq X$. Now, let us prove (ii). Assume by contradiction that $(\{z\} \oplus_m A) \subseteq X$. Since erosion (and consequently, Minkowski subtraction) is an increasing operator [10], $(\{z\} \oplus_m A) \ominus_m A \subseteq (X \ominus_m A)$. Note that the left side of the previous inclusion is a closing operator. Since closing is an extensive operator [10], we have that $X \subseteq (\{z\} \oplus_m A) \ominus_m A \subseteq (X \ominus_m A)$. Since $z \in X$, we have that $z \in (X \ominus_m A)$. But, this is an absurd, since by definition of erosion transform, $z \notin (X \ominus_m A)$. Therefore, $(\{z\} \oplus_m A) \not\subseteq X$. □

■

The following proposition guarantees the correctness of our morphological skeleton algorithm (see Algorithm 3).

Proposition 2.7: Let $X, A \subseteq \mathbb{Z}^d$ and $x \in X$. Then $\max\{f_X^A(x+v) : v \in A^t\} \leq f_X^A(x)$ if and only if $x \in S_X^A$.

D. Time Complexity Analysis of Skeleton Algorithm

All points of M_X have been visited by **Erosion_Transform** whose time complexity is $\Theta(|M_X| \cdot |A|)$. Thus, the time complexity of **Skeleton** is $\Theta(|M_X| \cdot |A| + |M_X|) = \Theta(|M_X| \cdot |A|)$. If $|A|$ is a constant, the algorithm **Skeleton** runs in linear time, that is, in $\Theta(|M_X|)$.

III. Jump-Miss Erosion Algorithm

In this section, we present a new and fast binary erosion algorithm based on string matching techniques that will be called *jump-miss erosion* algorithm.

A. String Matching Techniques

String matching algorithms are an important class of string algorithms that search for locations where one or several strings (also called patterns) are found within a larger string or text.

Let Σ be an alphabet (finite set). Formally, both pattern and searched text are concatenations of elements of Σ. A naïve algorithm for finding a pattern P within a text T is $O(|P| \cdot |T|)$. However, there are faster search algorithms based on preprocessing techniques which build data structures in order to reduce the size of the search space.

Some algorithms preprocess the input text (building a suffix tree [11] in linear time [12]), some others preprocess the pattern (KMP [13]), and others both pattern and text (searching for regular expressions [14]). In general, preprocessing techniques are linear-time algorithms proportional to the input size. Another interesting feature presented in string matching algorithms is the property to make *jumps* while the searching is performing. These jumps occur when a consecutive character sequence of the pattern *miss-matches* a character subsequence of the text (see, for example, the Boyer-Moore algorithm [15]).

By looking at the erosion definition (Eq. (1)), one can easily see that the erosion computation can be viewed as the search for translations of B into the input image X. Thus, in the context of our work, the alphabet is $\Sigma = \{0, 1\}$, corresponding to the pixel values in binary images; while patterns and texts correspond respectively to SEs and input images. In our case, the preprocessing step consists of computing the erosion transform and the morphological skeleton using a fixed adjacent set A. In the following, we state some results needed to guarantee the correctness of our algorithm. The next proposition gives a relationship between the value of an erosion transforms at a point h and maximal disks centered at h.

Proposition 3.1: Let $X, Y, A \subseteq \mathbb{Z}^d$ and $h \in X$. If $o \in A$ and $|A| > 1$, then $f_X^A(h) \leq f_Y^A(h)$ if and only if $D_X^A(h) \subseteq D_Y^A(h)$.

Proof: Let $m = f_X^A(h)$ and $n = f_Y^A(h)$. Then, by Proposition 2.6, $D_X^A(h) = (\{h\} \oplus_{m-1} A)$ and $D_Y^A(h) = (\{h\} \oplus_{n-1} A)$. ($\Rightarrow$) Let k a positive integer such that $m = k+n$. Thus, $D_Y^A(h) = (\{h\} \oplus_{m-1} A) \oplus_k A = D_X^A(h) \oplus_k A$. Since $o \in A$, the dilation by A (and consequently, the Minkowski addition) is an anti-extensive operator [1], and, therefore, $D_X^A(h) \subseteq D_Y^A(h)$. ($\Leftarrow$) Since $(\{h\} \oplus_{m-1} A) \subseteq (\{h\} \oplus_{n-1} A)$, $o \in A$ and $|A| > 1$, we have there exist an integer $k \geq 0$ such that $(\{h\} \oplus_{m-1} A) \oplus_k A = (\{h\} \oplus_{n-1} A)$ and, consequently, $m \leq n$. □

■

The next proposition guarantees that once f_B^A has been computed, $f_{B_h}^A$ can be easily computed by translating horizontally the grayscale image f_B^A by h, that is, given a point $s \in S_{B_h}^A$, $f_{B_h}^A(s) = f_B^A(s - h)$.

Proposition 3.2: Let $B, A \subseteq \mathbb{Z}^d$ and $h \in \mathbb{Z}^d$. Then $f_B^A(s) = f_{B_h}^A(s + h)$ for all $s \in S_B^A$.

The next Theorem gives necessary and sufficient conditions for a point $h \in \mathbb{Z}^d$ to be in $\varepsilon_B(X)$ using erosion transforms using certain adjacent sets.

Theorem 3.3: Let $A \subseteq \mathbb{Z}^d$ an adjacent set such that $o \in A$ and $|A| > 1$. Let $B \subseteq \mathbb{Z}^d$ and $h \in \mathbb{Z}^d$. Let $s \in S_{B_h}^A$. Then, $f_{B_h}^A(s) \leq f_X^A(s)$, if and only if $B_h \subseteq X$

Proof: ($\Rightarrow$) By Proposition 2.5, for each $p \in B_h$, there exists $s \in S_{B_h}^A$ such that $p \in D_{B_h}^A(s)$. Besides, by Proposition 3.1, $f_{B_h}^A(s) \leq f_X^A(s)$ if and only if $D_{B_h}^A(s) \subseteq D_X^A(s)$. By by the second statement of Proposition 2.4, $D_X^A(s) \subseteq X$, and, consequently, $p \in X$. Therefore, $B_h \subseteq X$. ($\Leftarrow$) Since $B_h \subseteq X$, $D_{B_h}^A(s) \subseteq D_X^A(s)$, and, by Proposition 3.1, we have $f_{B_h}^A(s) \leq f_X^A(s)$. □

■

As a corollary of Theorem 3.3, we have that it is not necessary to verify all points of B_h in order to check if $B_h \subseteq X$. It is sufficient to examine just the points in the skeleton $S_{B_h}^A$. This yields a faster algorithm to compute $\varepsilon_B(X)$ since $|S_{B_h}^A|$ is usually much smaller than $|B_h|$. However, it is still possible to build an even faster algorithm using jumps. The next two lemmas are used to prove the jump-miss theorem.

Lemma 3.4: Let $X, A \subseteq \mathbb{Z}^d$ and $z \in \mathbb{Z}^d$. If $e \in A^t$, then $f_X^A(z + e) \leq f_X^A(z) + 1$.

Proof: By contradiction. Assume that $f_X^A(z + e) > f_X^A(z) + 1$. By Proposition 2.6, we have that $D_X^A(z+e) = (\{z + e\} \oplus_{m-1} A)$ and $D_X^A(z) = \{z\} \oplus_{n-1} A$, where $m = f_X^A(z+e)$ and $n = f_X^A(z)$. By hypothesis, $m > n+1$. But, if $e \in A^t$ and $m > n + 1$, then $D_X^A(z) \oplus A \subseteq D_X^A(z + e)$, since $D_X^A(z) \oplus A = (\{z\} \oplus_{n-1} A) \oplus A = (\{z\} \oplus_{(n+1)-1} A) \subseteq \{z + e\} \oplus_{m-1} A = D_X^A(z + e) \subseteq X$. But, this is a contradiction, since $D_X^A(z)$ is a maximal disk of X, $D_X^A(z) \oplus A$ could not be contained in X. Then $f_X^A(z + e) \leq f_X^A(z) + 1$. □

■

Before presenting the jump-miss theorem, we need to define a total order over $\mathbb{Z}^d$ to make possible the jumps in the searching process. Let $\mathcal{C} = \{e_i : 1 \leq i \leq d\}$ denote the *canonical basis* of $\mathbb{Z}^d$, that is, e_i is the vector with 1 at the ith coordinate and 0 elsewhere. The vector $e_i \in \mathcal{C}$ defines a permutation π_i of the sequence $[1, 2, \ldots, d]$ by exchanging the numbers i and d in that sequence, that is, $\pi_i = [1, 2, \ldots, i-2, i-1, d, i+1, i+2, d-2, d-1, i]$. For example, $\pi_1 = [d, 2, 3, \ldots, d-2, d-1, 1]$. More formally,

for $j = 1, \ldots, d$,

$$\pi_i[j] = \begin{cases} d & \text{if } j = i; \\ i & \text{if } j = d; \\ j & \text{otherwise.} \end{cases} \quad (6)$$

The sequence π_i defines a total order $\leq_i$ over $\mathbb{Z}^d$ in the following way $(a_1, a_2, \ldots, a_d) \leq_i (b_1, b_2, \ldots, b_d)$ if and only if $\exists m > 0 : (\forall j < m, a_{\pi_i[j]} = b_{\pi_i[j]})$ and $(a_{\pi_i[m]} < b_{\pi_i[m]})$. For example, in $\mathbb{Z}^2$, for the rectangle R from $(1,1)$ to (m,n), we have $(1,1) \leq_2 (1,2) \leq_2 \ldots \leq_2 (1,n) \leq_2 (2,1) \leq_2 \ldots \leq_2 (2,n) \leq_2 \ldots \leq_2 (m,n)$ The order in which the points in R are visited in this case is known as *raster order*. Note that $e_2 = (0,1)$ is the horizontal axis.

Given a set $X = \{x_1, \ldots, x_m\} \subseteq \mathbb{Z}^d$, let $L_X^i = (x_{k_1}, \ldots, x_{k_m})$ denote the sequence of points of X such that $x_{k_j} \leq_i x_{k_{j+1}}$. If $z \in \mathbb{Z}^d$ such that $x_{k_1} \leq_i z <_i x_{k_m}$, the next element to z in X, denoted by $next_X(z) \in L_X^i$, is defined as $x_{k_j} \in X$ such that $x_{k_{j-1}} <_i z \leq_i x_{k_j}$. If $z = x_{k_m}$, then we define $next_X(z) \in L_X^i$ as λ.

Theorem 3.5 (Jump-Miss Theorem): Let $X, B \subseteq \mathbb{Z}^d$. Let $A \subseteq \mathbb{Z}^d$ such that $A^t \cap \mathcal{C} \neq \emptyset$ and $o \in A$. Let $s \in S_B^A$. If $f_X^A(s) < f_B^A(s)$, then there are at least $(f_B^A(s) - f_X^A(s))$ points $h \in \mathbb{Z}^d$ such that $B_h \not\subseteq X$.

Proof: Let $e \in A^t \cap \mathcal{C}$. By hypothesis, $f_X^A(s) < f_B^A(s)$, that is, there exists an integer $k > 0$ such that $f_X^A(s) + k = f_B^A(s)$. By definition, $f_X^A(s) \geq 0$ and, by Proposition 3.4, for $i = 0, 1, ..., k$, $f_X^A(s + i \cdot e) \leq f_X^A(s) + i$. Thus, $f_X^A(s + i \cdot e) \leq f_B^A(s)$. Besides, by Proposition 3.2, for any i, $f_B^A(s) = f_{B_{i \cdot e}}^A(s + i \cdot e)$. Then, for $i = 0, 1, \ldots, k-1$, $f_X^A(s + i \cdot e) < f_{B_{i \cdot e}}^A(s + i \cdot e)$. Since $k = f_B^A(s) - f_X^A(s)$, we have $f_B^A(s) - f_X^A(s)$ positions $h = i \cdot e$ such that $f_X^A(s + h) < f_{B_h}^A(s + h)$. By Theorem 3.3, for each h, $B_h \not\subseteq X$. □ ■

Note that the order in which the pixels are visited (and, consequently, the direction in which we have jumps) is defined by the vector $e \in \mathcal{C}$. Finally, in Algorithm 4, we present the jump-miss erosion algorithm.

B. Time Complexity Analysis of Jump-Miss Erosion Algorithm

The time complexity analysis of the algorithm depends on the data structure. In this work, we have chosen M_X, L_X and M_B, L_B for storing the sets X and B, respectively. We can use a linked list to store S_B^A in order to perform efficiently the comparison at Line 9. In addition, we use the ordered set L_X^i to store the points of X to perform efficiently the instructions at Lines 7, 14 and 18. With these optimizations, the time complexity, in the worst case, is $O(|M_B| \cdot |A| + |M_X| \cdot |A| + |S_B^A| \cdot |X|)$. If $|A|$ is a constant, then the time complexity of **Jump_Miss_Erosion** is $O(|M_B| + |M_X| + |S_B^A| \cdot |X|)$ which is much smaller than $O(|M_B| \cdot |M_X|)$, the time complexity for most known erosion implementation, since usually, $|S_B^A| \ll |M_B|$.

Algorithm 4 Jump_Miss_Erosion (X, B, E)

Input: an image X, a SE B such that $o \in B$, and an adjacent set A such that $o \in A$ and $A^t \cap \mathcal{C} \neq \emptyset$.
Output: the set $Z = X \ominus B$.

```
1:  /* Begin Preprocessing Step */
2:  (S_B^A, f_B^A) ← Skeleton (B, A);
3:  f_X^A ← Erosion_Transform (X, A);
4:  /* End Preprocessing Step */
5:  Z ← ∅;
6:  Let e_i ∈ A^t ∩ C;
7:  h ← The first element of L_X^i;
8:  while h ≠ λ do
9:    if (∃s ∈ S_{B_h}^A : f_B^A(s − h) > f_X^A(s)) /* miss-
      matching */ then
10:     /* jump */
11:     jump ← f_B^A(s − h) − f_X^A(s);
12:     h ← h + jump · e_i;
13:     if M_X(h) = 0 /* h ∉ X */ then
14:       h ← next_X(h);
15:     end if
16:   else
17:     Z ← Z ∪ {h};
18:     h ← next_X(h);
19:   end if
20: end while
21: return Z;
```

IV. Experimental Results

In this section, we will present some results obtained by the application of our algorithm to bi-dimensional binary images and compared it to other known erosion algorithms. We did two experiments: (i) one using square SEs with different sizes (in order to evaluate the performance of our algorithm using simple SEs); (ii) and the other one using various silhouettes[1] as structuring elements (see Fig. 3).

The input image used for both experiments is a 2500×2500 image and it has four sectors as you can see in Fig. 1. Two sectors are composed by four silhouettes (corresponding structuring elements), another sector by an image with salt and pepper noise with uniform distribution, and the last sector is composed by lines "equally" separated at slope about 150 degrees.

In all these experiments, the appropriate adjacent set chosen for our algorithm is $A = \{(0,0), (0,-1)\}$. Implementations for these and others experiments can be found at `http://score.ime.usp.br/~dandy/mestrado.php`.

[1]These silhouettes can be found at `http://www.imageprocessingplace.com/`

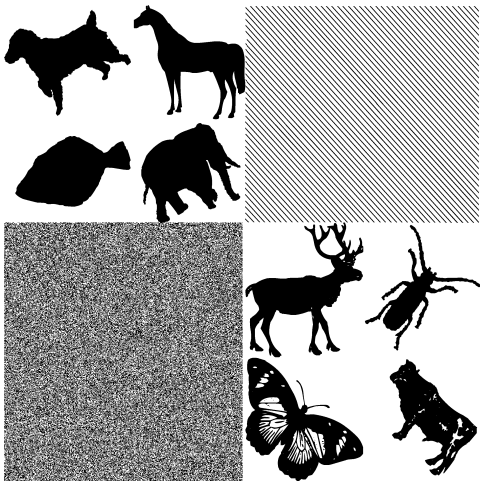

Fig. 1. The input image with dimension 2500×2500**.**

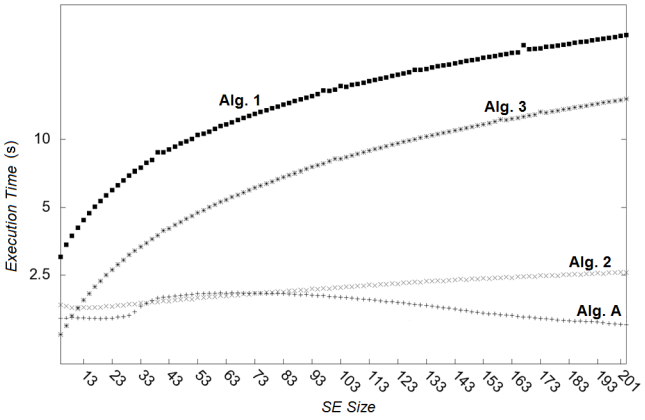

Fig. 2. Average time execution of the erosion algorithms using square SEs.

A. Square SEs

In order to show that our algorithm can be applied to images using simpler SEs such as squares, rectangles, and so forth, we have done experiments using square SEs with different sizes.

The erosion algorithms used in our experiment are the following: the first one is an algorithm that uses SE decomposition [16] (**Alg. 1**), the second one uses Run-Length Encoding (RLE) [17] (**Alg. 2**), the third one uses bit vector [18] with SE decomposition (**Alg. 3**), and, finally, our algorithm (**Alg. A**).

As you can see in Fig. 2, the execution time of the algorithms **Alg. 1** and **Alg. 3** had a significant influence as the SE size increases. In particular, **Alg. 3** seems to provide a better alternative when SE size is smaller than 10, which is a common situation in many applications. However, with the advance of technology, the increase number of high resolution images will require larger SEs, making **Alg. A** more appropriate for these situations.

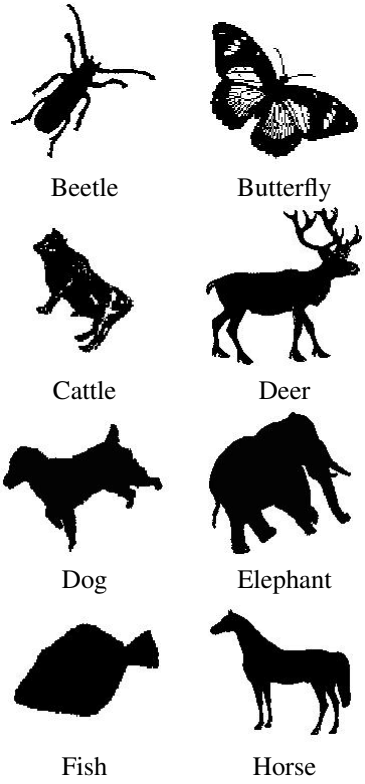

Fig. 3. Silhouettes used as SEs.

For algorithms **Alg. 2** and **Alg. A**, we observe that an increase in SE size has little effect on the execution time. Thus, **Alg. 2** and **Alg. A** have a good performance as the SE size increases. Moreover, there is a wide range of SE size where **Alg. 2** is slightly faster than **Alg. A**. This performance profile shows that our algorithm and **Alg. 2** are comparable. However, observe that **Alg. A** is the fastest from a certain SE size. This efficiency can be explained, since the more is $f_X^A(z)$, the larger are the jumps.

B. Arbitrary Shaped SEs

We have also compared our algorithm using arbitrary shaped SEs. For this experiment, we have used various silhouettes as structuring elements (see Fig. 3). The size of this silhouettes is around 600×600.

The erosion algorithms used in our experiment are the following: the first one is an algorithm that uses Run-Length Encoding (RLE) [17] (**Alg. 3**), the second one uses

Binary Decision Diagrams (BDD) [19] (**Alg. 4**), the third one uses bit vector [18] (**Alg. 5**), and, finally, our algorithm (**Alg. A**).

Table I shows the average time execution (in seconds) of the algorithms for ten repetitions using the silhouettes indicated in its first column as the SEs. The average time execution of the erosion algorithms marked by boldface corresponds to the fastest algorithm. The values '-' indicated in the table correspond to average time execution bigger than 2,000 seconds.

SEs	***Alg. 3***	***Alg. 4***	***Alg. 5***	***Alg. A***
beetle	10.53	36.14	1282.74	**2.64**
butterfly	33.37	3.59	–	**1.81**
cattle	10.42	5.28	–	**1.83**
deer	11.01	6.27	–	**1.72**
dog	6.64	2.71	–	**1.55**
elephant	6.48	21.19	–	**1.61**
fish	4.91	82.33	–	**1.84**
horse	5.85	63.05	–	**1.90**

TABLE I. The average time execution of the erosion algorithms.

As you can see in Table I, the average execution time of the algorithms **Alg. 3** and **Alg. 4** had a significant influence depending on the type of the SE. For algorithm **Alg. A**, we observe that these SEs have little effect on the average execution time. Thus, **Alg. A** has good performance for arbitrary SEs. As for the first experiment, this efficiency can be explained, since the more is $f_X^A(z)$, the larger are the jumps.

V. Conclusion

In this paper, we have proposed an efficient algorithm for binary erosion inspired by preprocessing techniques which is quite similar to those presented in many fast string matching algorithms (jumps and miss-matchings). We have also shown that the time complexity of this algorithm has clear advantages over some known erosion implementations. Tests comparing the execution time of some known erosion algorithms confirm the theoretical analysis and show that our algorithm has a good performance and it is a better option for erosion computation.

An important issue that remains for future research is to find the appropriate adjacent set A that minimizes S_B^A and, consequently, the execution time of the erosion algorithm. Other problem is to select the adjacent set A that maximizes the value of $f_X^A(s)$, where $s \in S_X^A$. Currently, we are investigating strategies to solve these problems using optimization techniques.

VI. Acknowledgments

This work was supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), and FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo).

References

[1] Dougherty, E.R., Lotufo, R.A.: Hands-on Morphological Image Processing. SPIE International Society for Optical Engine (2003)
[2] Birkhoff, G.: Lattice Theory. American Mathematical Society Colloquium Publications, Rhode Island (1967)
[3] Serra, J.: Image Analysis and Mathematical Morphology. Academic Press, New York (1982)
[4] Serra, J.: Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances. Academic Press (1988)
[5] Borgefors, G.: Distance Transformations on Digital Images. Computer Vision and Image Processing **51**(34) (1986) 344–371
[6] Chen, S., Haralick, R.M.: Recursive erosion, dilation, opening, and closing transforms. IEEE Transactions on Computers **4**(3) (March 1995) 335–345
[7] Lay, B.: Recursive algorithm in mathematical morphology. Acta Stereologica **6** (1987) 691–696
[8] Maragos, P., Schafer, R.: Morphological skeleton representation and coding of binary images. IEEE Transactions on Acoustics, Speech ans Signal Processing **34** (October 1986) 1228–1244
[9] Lantuejoul, C.: Skeletonization in quantitative metallography. In Haralick, R.M., Simon, J.C., eds.: Issues in Image Processing. Sijthoff and Noordhof (1980) 107–135
[10] Heijmans, H.J.A.M.: Morphological Image Operators. Academic Press, Boston (1994)
[11] Weiner, P.: Linear pattern matching algorithm. In: 14th Annual IEEE Symposium on Switching and Automata Theory, ACM/IEEE (1973) 1–11
[12] Ukkonen, E.: On-line construction of suffix trees. Algorithmica **3**(14) (1995) 249–260
[13] Knuth, D., James H. Morris, J., Pratt, V.: Fast pattern matching in strings. SIAM Journal on Computing **2**(6) (1977) 323–350
[14] Baeza-Yates, R.A., Gonnet, G.H.: Fast text searching for regular expressions or automaton searching on tries. Journal of the ACM (43) (1996) 915–936
[15] R. S. Boyer, J.S.M.: A fast string searching algorithm. Comm. ACM **20** (1977) 762–772
[16] Xu, J.: Decomposition of Convex Polygonal Morphological Structuring Elements into Neighborhood Subsets. IEEE Transactions on Pattern Analysis and Machine Inteligence **13**(2) (February 1991) 153–162
[17] Kim, W.J., Kim, S.D., Kim, K.: Fast Algorithms for Binary Dilation and Erosion Using Run-Length Encoding. ETRI Journal **27**(6) (December 2005) 814–817
[18] Bloomberg, D.S.: Implementation Efficiency of Binary Morphology (April 2002) http://www.leptonica.com/papers/binmorph.pdf.
[19] Madeira, H.M.F., Barrera, J., Jr., R.H., Hirata, N.S.T.: A New Paradigm for the Architecture of Morphological Machines: Binary Decision Diagrams. In: SIBGRAPI'99 - XII Brazilian Symposium of Computer Graphic and Image Processing, IEEE Computer Society (November 1999) 283–292