7 3 4

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 2, p. 49–50.
http://urlib.net/dpi.inpe.br/ismm@80/2007/06.16.00.48

Efficient binary erosion algorithm based on a string-matching-like technique

Anderson Fraiha Machado,
Ronaldo Fumio Hashimoto and
Alair Pereira do Lago

Universidade de São Paulo (USP), Brazil
{dandy,ronaldo,alair}@ime.usp.br

1. Introduction

Let E be a nonempty subset of $\mathbb{Z}^d$ and let $\mathcal{P}(E)$ denote the power set of E. Let $h \in \mathbb{Z}^d$ and $X \subseteq E$. The set $X_h = \{x+h : x \in X\}$ is the *translation* of X by h. Let $B \in \mathcal{P}(E)$. We define $\varepsilon_B : \mathcal{P}(E) \to \mathcal{P}(E)$ the *erosion* by B, also called *structuring element*, $\varepsilon_B(A) = \{h \in \mathbb{Z}^d : B_h \subseteq A\}$ for all $A \in \mathcal{P}(E)$. More details can be found in [4].

This work presents a new algorithm for binary morphological erosions inspired by a preprocessing technique which is quite similar to those presented in many fast string matching algorithms [1]. A time complexity analysis shows that this algorithm has clear advantages over the traditional and quite naive implementations which consist of passing a structuring element over the input image. Experimental results confirm this analysis and shows that this algorithm has a good performance and is a better option for erosions computations.

2. The new algorithm for erosion

This section introduces the proposed algorithm for binary morphological erosions.

2.1 Preprocessing

Let $x \in E$. We denote by $[x]_k$ the k^{th} dimension of the point x. Thus $x = ([x]_1, [x]_2, \ldots, [x]_d)$.

The first preprocessing step

Let $X \in \mathcal{P}(E)$ and $k \in \{1, 2, \ldots, d\}$. The first preprocessing step consists of using the k^{th} dimension of the space E to find a partition $\{P_1, P_2, \ldots, P_\ell\}$ of X, that has the following property: $x, y \in X$ are in the same subset of partition if, and only if, for all $j \neq k$, $[x]_j = [y]_j$. There exists an algorithm to find this partition in $O(|X|)$.

Let $x, y \in P_i$. The point x is *adjacent by dimension k*, or simply *adjacent*, to y if and only if $|[x]_k - [y]_k| = 1$. A nonempty subset $I = \{x_0, x_1, \ldots, x_n\} \subseteq P_i$ is an *interval of* X if, and only if, $\forall x_j \in I$ with $j < n$, x_{j+1} is adjacent by dimension k to x_j. An interval $I \subseteq P_i$ is *maximal* if, and only if, $\forall I' \subseteq P_i$, $I' \neq I$, we have that $I \not\subseteq I'$. The set of all maximal intervals of P_i is denoted by $\mathcal{I}_i$. The *set of all maximal intervals of* X is defined as $\mathcal{I}(X) = \{I \in \mathcal{I}_i : \ i = 1, 2, \ldots, \ell\}$.

The second preprocessing step

Let $X \in \mathcal{P}(E)$. The second preprocessing step consists of finding the set $\mathcal{I}(X)$. If we use a data structure (e.g., multidimensional array) that allows us to verify if an element $x \in E$ is an element of P_i in time $O(1)$, there exists an algorithm that builds $\mathcal{I}_i$ in time $O(|P_i|)$. Thus, since $\{P_1, \ldots, P_\ell\}$ is a partition of X, there exists an algorithm to find $\mathcal{I}(X)$ with complexity time $O(|X|)$.

For each interval $I \in \mathcal{I}(X)$, its *extremities* are the points $p_{\min}(I) \in I$ and $p_{\max}(I) \in I$ such that $[p_{\min}(I)]_k \leq [x]_k \leq [p_{\max}(I)]_k$ for all $x \in I$. Notice that for each point $x \in X$, there exists only one interval $I \in \mathcal{I}(X)$ that contains x. Let $X \in \mathcal{P}(E)$. The *density* of x with respect to X, denoted by $\Delta_X(x)$, is defined as (see Figure 1):

$$\Delta_X(x) = \begin{cases} [x]_k - [p_{\min}(I)]_k & \text{if } x \in X \\ -1 & \text{otherwise} \end{cases}$$

where $I \in \mathcal{I}(X)$ is the only interval that contains x.

The third preprocessing step

Let $X \in \mathcal{P}(E)$. The third preprocessing step consists of computing the densities of all $x \in X$. Given $I \in$

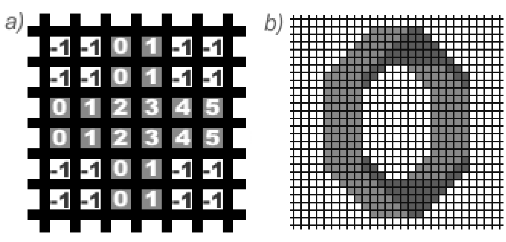

Figure 1. (a) A structuring element B with its respective densities. (b) An input image A; the darker gray color indicates points $x \in A$ such that $\Delta_A(x) \geq \Delta_B(b_{\max})$.

Proceedings of the 8th International Symposium on Mathematical Morphology,
Rio de Janeiro, Brazil, Oct. 10–13, 2007, MCT/INPE, v. 2, p. 49–50.
http://urlib.net/dpi.inpe.br/ismm@80/2007/06.16.00.48

$\mathcal{I}(X)$, it is possible to implement an algorithm for computing the densities of all $x \in I$ in time $O(|I|)$. Thus, since $\mathcal{I}(X)$ is a partition set of X, there exists an algorithm for finding the density for all points of X in $O(|X|)$.

We will denote by $x_{\max}$ the point in X such that its density is maximum. It is obvious that we can find this point in time $O(|X|)$.

For each interval $I \in \mathcal{I}(X)$, the *shell* of I, denoted by $\zeta(I)$, is the point $\zeta(I) = p_{\max}(I)$ (see Figure 1).

Let $A, B \in \mathcal{P}(E)$. If $\Delta_B(\zeta(I)) \leq \Delta_A(\zeta(I)), \forall I \in \mathcal{I}(B)$, then $B \subseteq A$. Let $X \in \mathcal{P}(E)$ and $h \in \mathbb{Z}^d$. For each $I \in \mathcal{I}(X_h)$ there exists $I' \in \mathcal{I}(X)$ such that $\zeta(I) = \zeta(I') + h$ and $\Delta_{X_h}(\zeta(I)) = \Delta_X(\zeta(I'))$.

2.2 The erosion algorithm

Based on the previous definitions and properties, we present the proposed erosion algorithm.

1: **Erosion** (A, B, k)
2: **Input**: $A, B \in \mathcal{P}(E)$ and $k \in \{1, 2, \ldots, d\}$.
3: **Output**: $\varepsilon_B(A)$.
4: $\varepsilon_B(A) \leftarrow \varnothing$;
5: Let $b_{\max} \in B$ /* *that is,* $\Delta_B(b_{\max})$ *is maximum*/
6: **for all** $a \in A : \Delta_A(a) \geq \Delta_B(b_{\max})$ **do**
7: $\quad h = a - b_{\max}$;
8: $\quad$ **if** $\Delta_{B_h}(\zeta(I)) \leq \Delta_A(\zeta(I)), \forall I \in \mathcal{I}(B_h)$ **then**
9: $\quad\quad \varepsilon_B(A) \leftarrow \varepsilon_B(A) \cup \{h\}$;
10: $\quad$ **end if**
11: **end for**
12: **return** $\varepsilon_B(A)$;

3. Complexity analysis

Let us denote $\varphi(A, b_{\max})$ the number of points $a \in A$ such that $\Delta_A(a) \geq \Delta_B(b_{\max})$ (see Figure 1). Basically, $\varphi(A, b_{\max})$ is the number of times the condition at Line 6 is satisfied. Thus, the number of points of A that does not satisfy the condition at Line 6 is $|A| - \varphi(A, b_{\max})$. On the other hand, the complexity time for verifying the condition at Line 8 is $O(|\mathcal{I}(B)|)$ and, since this line is executed $\varphi(A, b_{\max})$ times, the complexity time of the algorithm is $O(|\mathcal{I}(B)| \cdot \varphi(A, b_{\max}))$. Since the running time for preprocessing A and B in order to compute the initial partition set, the maximal interval sets and the densities for all points of these sets is $O(|A| + |B|)$, the overall complexity time for computing $\varepsilon_B(A)$ is $O(|B| + |A| + |\mathcal{I}(B)| \cdot \varphi(A, b_{\max}))$.

This analysis shows that the proposed algorithm has clear advantages over the quite naïve implementations which have complexity time $O(|A| \cdot |B|)$ and consist of passing a structuring element over the input image.

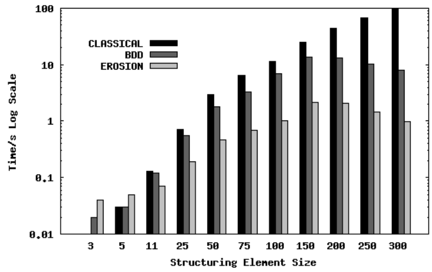

Figure 2. Average execution time among all algorithms using a PC with 3.0 GHz CPU and 1 Gbyte RAM.

4. Results and discussion

In this section, we present some experimental results of the proposed algorithm for dimension $d = 2$ and $k = 2$. To show its performance, we compared the execution time among the **CLASSICAL** (naïve implementation) and the **BDD** (based on Binary Decision Diagram [3]) algorithms. All algorithms for binary erosion have been executed on a pentium IV workstation running Linux operating system.

In our experiments we have used squares, diamonds and disks of dimension n ranging from 3 to 300 as structuring elements. As input images, we have used binary images[1] taken from a digital image processing database[2] used in [2].

The execution time of all algorithms is presented in Figure 2. These experimental results confirm the complexity analysis and shows that this algorithm has a good performance and is a better option for erosions computations.

This is still an ongoing research and as a future work, we plan to compare our algorithm with other erosion implementations known in the literature.

References

[1] D. Knuth, J. H. Morris Jr, and V. Pratt, *Fast Pattern Matching in Strings*, SIAM Journal on Computing **6** (1977), no. 2, 323–350.

[2] L. J. Latecki and R. Lakämper and U. Eckhardt, *Shape Descriptors for Non-rigid Shapes with a Single Closed Contour*, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2000, pp. 424–429.

[3] L. Robert and G. Malandain, *Fast Binary Image Processing Using Binary Decision Diagrams*, Computer Vision and Image Understanding **72** (1998), no. 1, 1–9.

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

[1] MPEG7 CE Shape-1 Part B
[2] http://www.imageprocessingplace.com/