Capítulo 3

Padrões de Algoritmos da Erosão

No capítulo anterior foi apresentada a teoria de operadores elementares caracterizados por funções estruturantes. Como exemplo, foram definidos operadores no reticulado das imagens binárias e no reticulado das imagens em níveis de cinza. Uma importante propriedade desses operadores é a decomposição de função estruturante.

Dando continuidade aos estudos de operadores morfológicos, será apresentada neste capítulo a erosão caracterizada por funções estruturantes focando as implementações. Assim, serão usados três padrões de algoritmos: paralelo, seqüencial e por propagação.

Existem resultados na morfologia matemática para a decomposições ótimas de funções estruturantes [HAS00]. Porém, serão apresentadas nesta tese apenas algumas decomposições para a viabilidade da transformada de distância usando erosões.

No próximo capítulo serão classificados os principais algoritmos da transformada de distância usando as erosões paralela, seqüencial e por propagação definidas a seguir.

3.1 Padrão paralelo

Será apresentado nesta seção o padrão paralelo, onde a ordem de varredura na imagem não influencia no resultado obtido.

3.1.1 Erosão paralela

Será apresentada a seguir a caracterização da erosão morfológica por função estruturante no reticulado $K^{\mathbb{E}}$ das imagens em níveis de cinza.

A erosão por uma função estruturante, definida na Equação 2.5, pode ser reescrita pelo Algoritmo 4.

Algoritmo 4 Erosão paralela

$\varepsilon : K^{\mathbb{E}} \times \mathbb{Z}^B \longrightarrow K^{\mathbb{E}}$
$\quad \varepsilon(f, b) = \varepsilon_b(f)$

$\forall x \in \mathbb{E}$ // em paralelo
$\quad \varepsilon_b(f)(x) = \bigwedge_{\forall y \in B_x \cap \mathbb{E}} \{f(y) \dot{-} b(y - x)\};$

Observe que este algoritmo é genérico, isto é, independe da dimensão e do tipo da imagem.

A notação utilizada para representar os algoritmos neste documento é a descrita na Seção 2.5. Esta notação é semelhante à notação Z e é possível gerar código a partir destes algoritmos. Isso será descrito com detalhes no Apêndice A – *Ambiente mmil*.

3.1.2 Decomposição paralela de função estruturante

A aplicação direta do Algoritmo 4 por uma função estruturante "grande"[1] b_G, apresentado na Subseção 3.1.1, é ineficiente em máquinas seqüenciais. Uma solução é a *decomposição de função estruturante* grande em funções estruturantes "pequenas", como apresentada mais adiante nesta seção. Para justificar esta afirmação, considere as próximas definições.

Sejam $B_i \subseteq \mathbb{E} \oplus \mathbb{E}$ contendo a origem e $b_i \in \mathbb{Z}^{B_i}$, onde $i = 1, \ldots, k$. Então a *soma de Minkowski em níveis de cinza* de b_i por b_j [Ser82, SM92] é definida como [2]: $\forall x \in B_i \oplus B_j$,

$$(b_i \oplus b_j)(x) = \max\{b_i(y) + b_j(x - y) : \ y \in (B_j + x)\}, \tag{3.1}$$

onde $j = 1, \ldots, k$ e $B_i \oplus B_j$ é a *soma de Minkowski* em conjunto[3] (imagens binárias), veja Figura 3.1.

Como conseqüência, é possível fazer a *soma de Minkowski generalizada em níveis de cinza* como segue:

$$b_G = b_1 \oplus \cdots \oplus b_k, \tag{3.2}$$

onde $b_1, \cdots, b_k$ são funções que decompõem b_G. Quando existe um b tal que,

$$b_G = \underbrace{b \oplus \cdots \oplus b}_{k \ vezes}, \tag{3.3}$$

[1] Como apresenado no Seção 1.2, dizemos função estruturante "grande" (por exemplo, com o tamanho do domínio equivalente a duas vezes o da imagem a ser processada) e função estruturante "pequena" (por exemplo, 3×3, 1×3 ou 1×2) que decompõe a função estruturante "grande".

[2] Observe que nesta parte B_i não é a translação de B por i, mas apenas a notação usual de um conjunto qualquer.

[3] Note que é usado o mesmo símbolo para soma de Minkowski para imagens binárias (representada por letras maiúsculas) e para imagens em níveis de cinza (representadas por letras minúsculas).

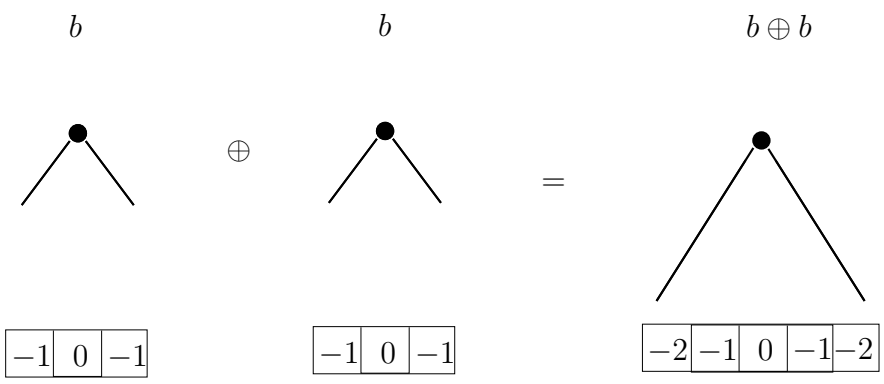


Figura 3.1: Soma de Minkowski em níveis de cinza, onde as origens estão no centro de cada função estruturante.

é escrito simplesmente $b_G = kb$.

Uma propriedade da erosão é [Ser82, SM92]:

$$\varepsilon_{b_G}(f) = \varepsilon_{b_k}(\cdots(\varepsilon_{b_1}(f))\cdots). \tag{3.4}$$

Note que é mais eficiente trabalhar com a decomposição de b_G em erosões paralelas. Isto também se aplica para os padrões seqüenciais e por propagação, como serão apresentados nas próximas seções. Por exemplo, seja $b_G = kb_i$ de dimensão 3×3 e f de dimensão $n \times n$. Então $\varepsilon_{b_G}(f)$ requer

$$(2k+1)(2k+1)n^2 = (2k+1)^2n^2$$

acessos à memória, enquanto $\varepsilon_{b_k}(\cdots(\varepsilon_{b_1}(f))\cdots)$ requer $9kn^2$ acessos.

A Figura 3.2 ilustra um exemplo de aplicação da erosão paralela de uma imagem f por uma função estruturante b.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

f

$$\begin{matrix} -1 & -1 & -1 \\ -1 & \mathbf{0} & -1 \\ -1 & -1 & -1 \end{matrix}$$

b

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$\varepsilon_b(f)$

Figura 3.2: Ilustração da erosão paralela $\varepsilon_b(f)$ da imagem de entrada f pela função estruturante b, com origem no centro.

3.2 Padrão seqüencial

Será apresentado nesta seção o *padrão seqüencial*. A principal característica deste padrão é a sua eficiência pois basta varrer uma imagem duas vezes para conseguir bons resultados. Este tipo de algoritmo é conhecido na literatura desde a década de 60 [RP66, Dia69, Dan80, YTF81, WB88, WB92, WHCR95, BHJ97, D'O01]. Como foi visto na seção anterior, nos algoritmos paralelos os pixels são processados independentemente da ordem de varredura da imagem, dependendo apenas dos valores dos pixels da imagem de entrada e da vizinhança usada. Nos algoritmos seqüenciais a imagem de saída depende não somente dos pixels da imagem de entrada, mas também dos valores calculados anteriormente e da ordem de varredura na imagem.

3.2.1 Erosão seqüencial

Será apresentada nesta subseção a erosão morfológica caracterizada por funções estruturantes no reticulado $K^{\mathbb{E}}$ das imagens em níveis de cinza. Os operadores que serão apresentados são diferentes daqueles vistos até agora no que diz respeito à ordem de varredura da imagem e na forma de decompor a função estruturante.

Algoritmos seqüenciais podem ser classificados conforme a ordem de varredura dos pixels acessados de uma imagem. As duas ordens mais comuns são as *raster* e *anti-raster*, como exemplo, veja Figuras 3.3a e 3.3b, respectivamente[4].

Considere uma imagem f com domínio $\mathbb{E} \subset \mathbb{Z} \times \mathbb{Z}$ de dimensões $m \times n = |\mathbb{E}|$, onde m é a *largura* ou o número de colunas da imagem e n é a *altura* ou o número de linhas da imagem. Um *pixel* em $\mathbb{E}$ é denotado pelo par ordenado

[4]D'Ornellas chamou esta varredura *raster* e *anti-raster* e varredura horizontal, definindo também a varredura vertical [D'O01].

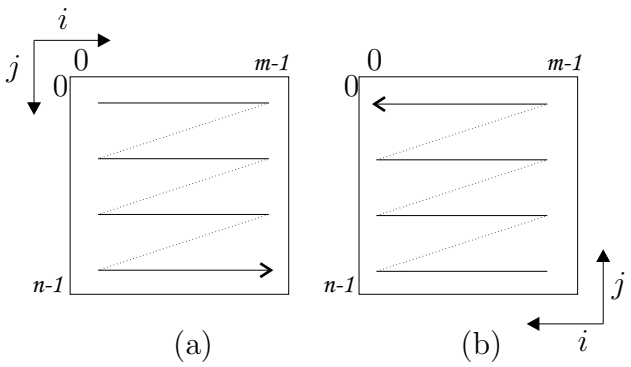


Figura 3.3: Ordem de varredura em uma imagem de duas dimensões: (a) *raster* horizontal e (b) *anti-raster* horizontal.

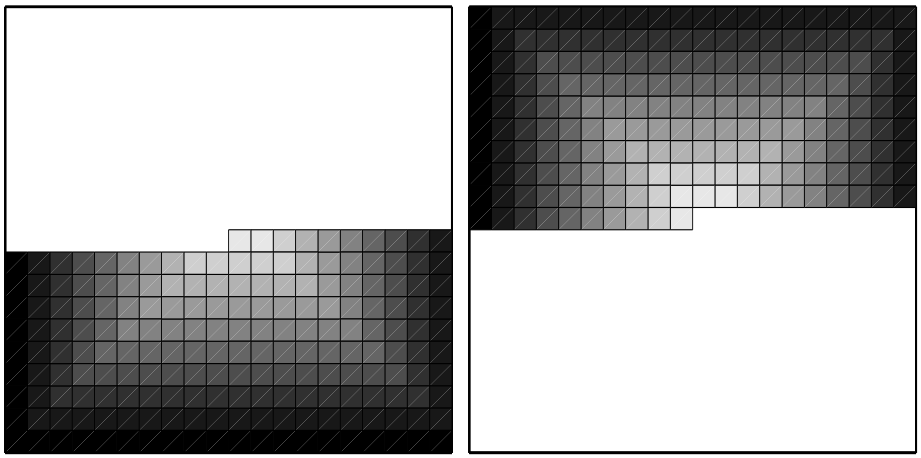
(i, j), onde $0 \leq i \leq m-1$ e $0 \leq j \leq n-1$.

No caso de uma varredura horizontal em uma imagem f, a ordem *raster* é definida como uma visita aos pixels de f da esquerda para a direita e de cima para baixo, isto é, $\{(0,0),(0,1),\cdots,(0,n-1),\ (1,0),(1,1),\cdots,(1,n-1),\ \cdots,(m-1,0),(m-1,1),\cdots,(m-1,n-1)\}$, veja Figura 3.3a. Esta seqüencia é definida como $S^+ = \{0,\cdots,mn-1\}$. A varredura na ordem *anti-raster* em f é definida na direção inversa, da direita para a esquerda e de baixo para cima, isto é, $\{(m-1,n-1),(m-1,n-2),\cdots,(m-1,0),\cdots,(0,n-1),(0,n-2),\cdots,(0,0)\}$. Esta seqüencia é definida como $S^- = \{mn-1,\cdots,0\}$, veja Figura 3.3b.

Existe uma bijeção $\ell^+ : \mathbb{E} \rightarrow S^+$, definida por $\ell^+(i,j) = in + j$, onde $(i,j) \in \mathbb{E}$ e n é a largura da imagem. Uma função inversa de ℓ^+ é dada por $(\ell^+)^{-1}(r) = (\lfloor r/n \rfloor, r\ rem\ n)$, onde $r \in \{1,2,\ldots,|S^+|\}$, $\lfloor r/n \rfloor$ é a divisão inteira e $r\ rem\ n$ é o resto da divisão de r por n, respectivamente.

Dado um elemento estruturante $B \subseteq \mathbb{E} \oplus \mathbb{E}$ com origem, seja B^+ uma vizinhança para a ordem *raster* e seja B^- uma vizinhança para a ordem *anti-raster*. Considerando o centro de B como o centro dos eixos de coordenadas cartesianas $(i,j) \in \mathbb{E}$, onde i é a abscissa (posição para direita) e j é a ordenada (posição para baixo), como ilustrados na Figura 3.3a, seja: $B^+ =$

$\{(i,j) \in B \mid j > 0$ ou se $j = 0$ então $i \geq 0\}$. Similarmente $B^- = \{(i,j) \in B \mid j < 0$ ou se $j = 0$ então $i \leq 0\}$. Note que $B^+ \cup B^- = B$. Similarmente, igual decomposição pode ser aplicada para uma função estruturante $b \in \mathbb{Z}^B$, isto é, $b^+ \in \mathbb{Z}^{B^+}$ e $b^- \in \mathbb{Z}^{B^-}$. Um exemplo desta decomposição é mostrada na Figura 3.4. Isto significa que qualquer função estruturante com domínio em $\mathbb{E} \oplus \mathbb{E}$ com origem pode ser decomposta em funções estruturantes *raster* e *anti-raster*.



b_G^+ b_G^-

Figura 3.4: Exemplos de funções estruturantes utilizadas nas varreduras *raster* e *anti-raster*.

Seja $\varepsilon_{b^+}^+(f)$ a *erosão seqüencial na ordem raster*, como definida na Equação 2.5, porém colocando os resultados parciais na própria função f. De forma mais precisa: para $x \in \mathbb{E}$ na ordem *raster*,

$$\varepsilon_{b^+}^+(f)(x) = \min\{\varepsilon_{b^+}^+(f)(y) \dot{-} b(y-x) : y \in B_x^+ \cap \mathbb{E}\}. \tag{3.5}$$

Analogamente, seja $\varepsilon_{b^-}^-(f)(x)$ a *erosão seqüencial na ordem raster* de f, definida como: para $x \in \mathbb{E}$ na ordem *anti-raster*,

$$\varepsilon_{b^-}^-(f)(x) = \min\{\varepsilon_{b^-}^-(f)(y) \dot{-} b(y-x) : y \in B_x^- \cap \mathbb{E}\}. \tag{3.6}$$

Baseados nas Equações 3.5 e 3.6 acima, serão apresentados os algoritmos da *erosão seqüencial na ordem raster* e da *erosão seqüencial na ordem anti-raster*, Algoritmos 5 e 6, respectivamente. Nestes algoritmos os conjuntos *seq* S^+ e *seq* S^- representam as varreduras *raster* e *anti-raster*, respectivamente. De forma análoga é definida a dilatação seqüencial e por este motivo não será apresentada aqui.

Algoritmo 5 Erosão seqüencial na ordem *raster*

$\varepsilon^+ : K^{\mathbb{E}} \times K^{B^+} \longrightarrow K^{\mathbb{E}}$
$\quad \varepsilon^+(f, b^+) = \varepsilon^+_{b^+}(f)$

$\varepsilon^+_{b^+}(f) = f;$
$\forall x \in seq\ S^+$ // seqüencial *raster*
$\quad \varepsilon^+_{b^+}(f)(x) = \bigwedge_{\forall y \in B^+_x \cap \mathbb{E}} \{\varepsilon^+_b(f)(y) \dot{-} b^+(y - x)\};$

A Figura 3.5 ilustra uma iteração intermediária da erosão seqüencial na ordem *raster*, $\varepsilon^+_{b^+}(f)(x)$, da imagem de entrada f pela função estruturante b^+. O pixel x, que está sendo processado, é o em negrito e está recebendo o valor $\mathbf{1} = \bigwedge\{0\dot{-}(-1), 1\dot{-}(-1), 2\dot{-}(-1), 0\dot{-}(-1), 5\dot{-}(0)\} = \varepsilon^+_{b^+}(f)(x)$.

Algoritmo 6 Erosão seqüencial na ordem *anti-raster*

$\varepsilon^- : K^{\mathbb{E}} \times K^{B^-} \longrightarrow K^{\mathbb{E}}$
$\quad \varepsilon^-(f, b^-) = \varepsilon^-_{b^-}(f)$

$\varepsilon^-_{b^-}(f) = f;$
$\forall x \in seq\ S^-$ // seqüencial *anti-raster*
$\quad \varepsilon^-_{b^-}(f)(x) = \bigwedge_{\forall y \in B^-_x \cap \mathbb{E}} \{\varepsilon^-_b(f)(y) \dot{-} b^-(y - x)\};$

$$
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 5 & 5 & 0 & 0 \\
0 & 0 & 5 & 5 & 5 & 0 & 0 \\
0 & 0 & 5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

f

$$
\begin{array}{ccc}
 & \mathbf{0} & -1 \\
-1 & -1 & -1 \\
 & b^+ &
\end{array}
$$

$$
\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 0 & 0 \\
0 & 0 & \mathbf{1} & 5 & 5 & 0 & 0 \\
0 & 0 & 5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

$\varepsilon^+_{b^+}(f)$

Figura 3.5: Ilustração de uma iteração intermediária da erosão seqüencial *raster*, $\varepsilon^+_{b^+}(f)(x)$, da imagem de entrada f pela função estruturante b^+, com origem em negrito. O pixel x, que está em negrito com valor $\mathbf{1}$, é o que está sendo processado.

3.2.2 Decomposição seqüencial de função estruturante

De forma análoga ao ocorrido nos algoritmos paralelos, a aplicação direta da erosão por uma função estruturante grande b_G é ineficiente. Contudo, é possível fazer a *decomposição seqüencial da função estruturante* b_G para obter implementações rápidas, como apresentado a seguir.

Um resultado de Rosenfeld e Pfaltz [RP66] é apresentado no teorema a seguir:

Teorema 3.1 *Aplicar um operador seqüencial com uma vizinhança local em uma imagem f é equivalente a aplicar uma seqüência de transformações paralelas com igual vizinhança local em f.* ■

A prova deste resultado pode ser vista no apêndice do trabalho [RP66] e algumas aplicações serão apresentadas a seguir, mais especificamente, serão apresentados casos onde se obtém a equivalência entre as erosões paralelas e seqüenciais.

Primeiro caso: $\varepsilon_{b_G}(f) = \varepsilon^-_{b^-}(f) \wedge \varepsilon^+_{b^+}(f)$

Seja b uma função estruturante qualquer contendo a origem b_o com o valor zero. Para o caso de b_o ser diferente de zero, basta subtrair b de b_o ($b = b - b_o$) e nos dois lados da igualdade acima calcular a erosão por b_o:

$$\varepsilon_{b_o}(\varepsilon_{b_G}(f)) = \varepsilon_{b_o}(\varepsilon^-_{b^-}(f) \wedge \varepsilon^+_{b^+}(f)).$$

Seja $b = b^+ \vee b^-$ a decomposição de b nas ordens *raster* b^+ e *anti-raster* b^-. Observe que as origens de b^+ e b^- também têm valores zeros.

Seja uma *função estruturante estável* uma função que não muda dentro de uma janela finita:

$$kb = (k+1)b, \tag{3.7}$$

por exemplo, dentro da janela $k \times k$.

Sejam b^+ e b^- funções estruturantes estáveis, $f \in K^{\mathbb{E}}$ e

$$b_G = k(b^+ \vee b^-) = kb^+ \vee kb^-, \tag{3.8}$$

então,

$$\varepsilon_{b_G}(f) = \varepsilon_{kb^+ \vee kb^-}(f) = \varepsilon_{kb^+}(f) \wedge \varepsilon_{kb^-}(f) = \varepsilon^-_{b^-}(f) \wedge \varepsilon^+_{b^+}(f). \tag{3.9}$$

Assim, a erosão paralela $\varepsilon_{b_G}(f)$ pode ser obtida pelos algoritmos *raster* e *anti-raster* $\varepsilon^-_{b^-}(f)$ e $\varepsilon^+_{b^+}(f)$, para uma função estruturante qualquer. A Figura 3.6 ilustra esta equivalência para uma função estruturante unidimensional. Na figura são ilustradas a imagem de entrada f de tamanho 1×10; a função estruturantes b de tamanho 1×5 com origem no centro; a erosão *raster* $\varepsilon^+_{b^+}(f)$ e b^+; a erosão *anti-raster* $\varepsilon^-_{b^-}(f)$ e b^-; e a equivalência $\varepsilon_{kb^+ \vee kb^-}(f) = \varepsilon^+_{b^+}(f) \wedge \varepsilon^-_{b^-}(f)$, com $b_G = kb^+ \vee kb^-$. Foi utilizado neste exemplo $k = 10$.

A desvantagem desta equivalência está nas funções estruturantes bidimensionais, onde ocorre o inconveniente ilustrado na Figura 3.7. Observe que na união $b_G = 2b^- \vee 2b^+$ existem dois espaços em branco, que possuem valores $-\infty$. Este problema se propaga com o aumento das somas de Minkowski. Assim, não é possível calcular a TD utilizando esta equivalência.

Outra ilustração da equivalência para o caso bidimensional é apresentada na Figura 3.8. Na figura são ilustradas a imagem de entrada f de tamanho 10×10, com os níveis de cinza gerados de forma aleatória[5]; a função estruturantes b de tamanho 3×3, também com níveis de cinza gerados de forma aleatória e com origem no centro; a erosão *raster* $\varepsilon^+_{b^+}(f)$; a erosão *anti-raster* $\varepsilon^-_{b^-}(f)$; e a equivalência $\varepsilon_{kb^+ \vee kb^-}(f) = \varepsilon^+_{b^+}(f) \wedge \varepsilon^-_{b^-}(f)$, com $b_G = kb^+ \vee kb^-$. Foi utilizado neste exemplo $k = 27$.

[5]Observe que à direita de cada imagem possui uma escala dos níveis de cinza existentes em cada imagem.

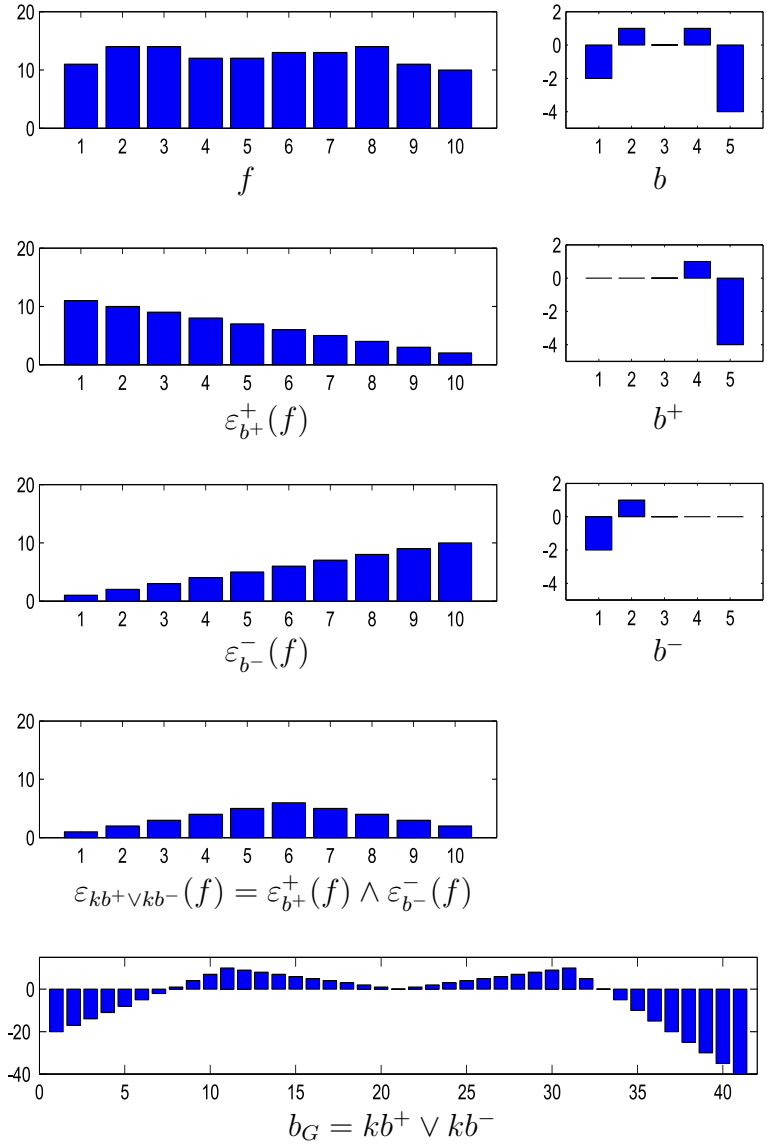


Figura 3.6: Ilustração da equivalência entre algoritmos paralelos e seqüenciais para funções estruturantes unidimensionais.

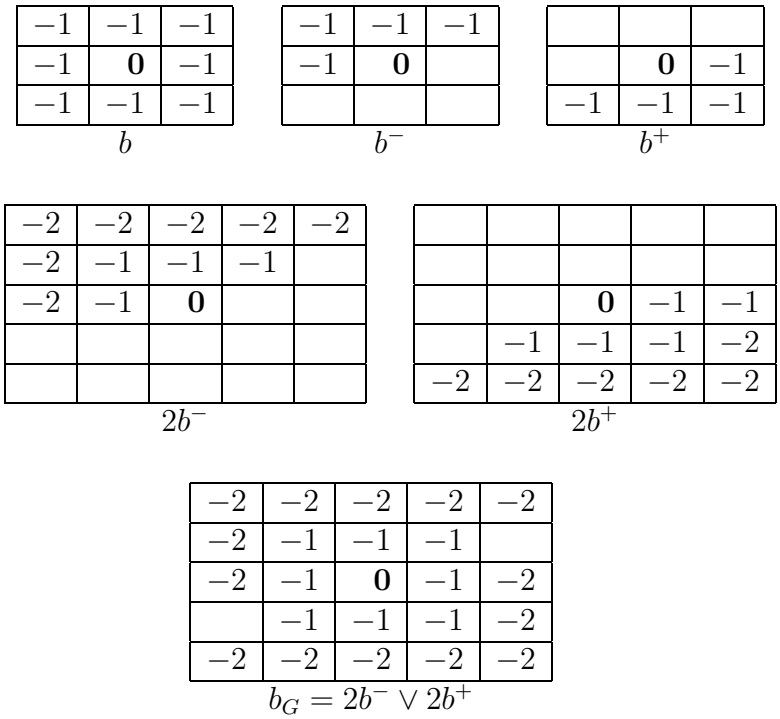


Figura 3.7: Ilustração da patologia ocorrida no primeiro caso da equivalência das erosões paralelas e seqüenciais usando $b_G = 2b^- \vee 2b^+$. Os espaços assumem $-\infty$.

Como sugestões para pesquisas futuras, seria interessante estudar a relação entre k e as dimensões e níveis de cinza de f e b, e não simplesmente assumir k suficientemente grande ou tendendo para o infinito para obter a equivalência entre erosões paralelos e seqüenciais[6].

Segundo caso: $\varepsilon_{b_G}(f) = \varepsilon^-_{b^-}(\varepsilon^+_{b^+}(f))$

Neste segundo caso, só existe uma função estruturante paralela equivalente para *funções estruturantes estáveis* (que não mudam dentro de uma janela finita). Quando isto ocorre, também vale a igualdade:

$$b_G = kb = kb^+ \oplus kb^-. \tag{3.10}$$

Como estudo de casos de funções estruturantes estáveis, veja a seguinte definição.

Seja $2p < q < p \leq 0$ conforme a função estruturante b mostrada na Figura 3.9 [Bor86], então

$$\varepsilon^+_{b^+}(f) = \varepsilon_{kb^+}(f) \quad e \quad \varepsilon^-_{b^-}(f) = \varepsilon_{kb^-}(f).$$

Portanto, é possível obter um operador paralelo equivalente a dois operadores seqüenciais. Por exemplo, seja k a maior distância possível numa imagem $f \in [0,k]^{\mathbb{E}}$. Se $b_G = kb$, então

$$\varepsilon_{kb}(f) = \varepsilon_{k(b^+ \oplus b^-)}(f) = \varepsilon_{kb^+ \oplus kb^-}(f) = \varepsilon_{kb^-}(\varepsilon_{kb^+}(f)) = \varepsilon^-_{b^-}(\varepsilon^+_{b^+}(f)). \tag{3.11}$$

Assim, a erosão paralela $\varepsilon_{b_G}(f)$ pode ser obtida pelos algoritmos *raster* e *anti-raster* $\varepsilon^-_{b^-}(\varepsilon^+_{b^+}(f))$, para uma função estruturante definida pela Figura 3.9.

[6]Um trabalho interessante que estuda as condições para que dilatações e erosões em espaço-escala morfológico sejam idempotentes foi realizado por Leite e Teixeira [LT00]. Neste trabalho também existe um resultado para o número de iterações necessárias, entre escalas, para que a idempotência seja calcula. Este resultado seria o ponto de partida para determinar a relação entre k e as dimensões e níveis de cinza de f e b.

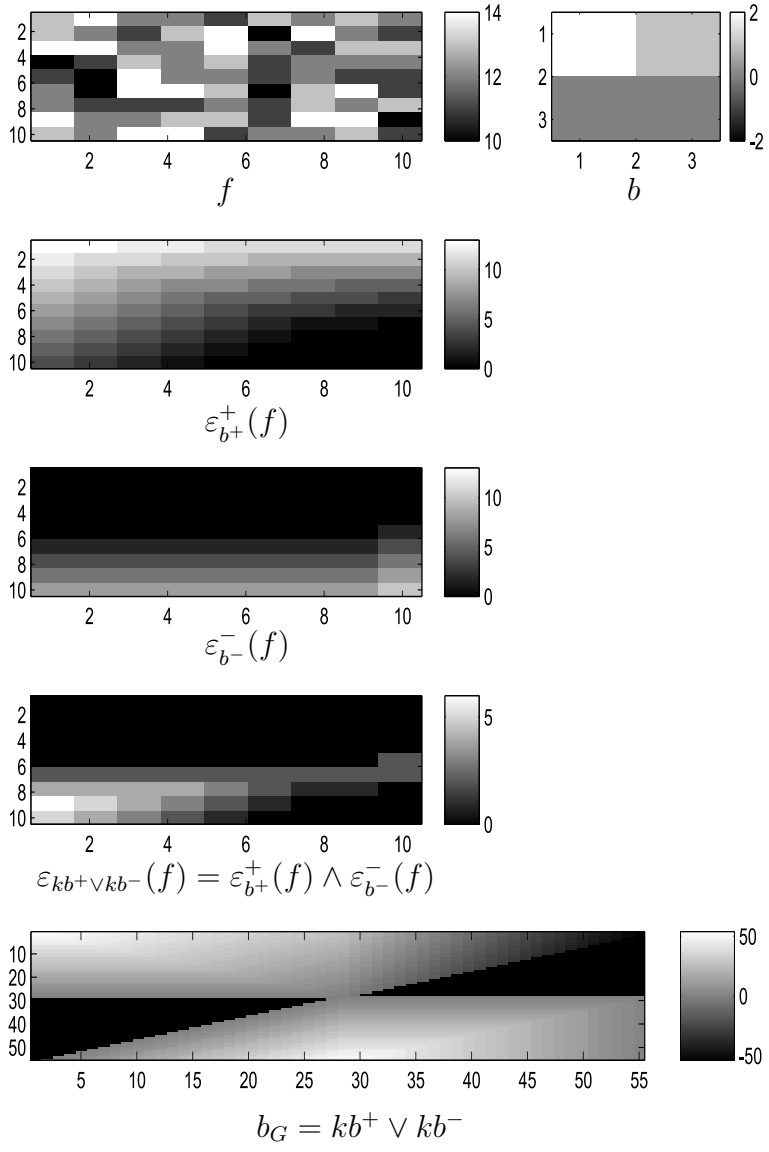


Figura 3.8: Ilustração da equivalência entre algoritmos paralelos e seqüenciais para funções estruturantes bidimensionais.

q	p	q
p	**0**	p
q	p	q

b

Figura 3.9: Função estruturante b para a Equação 3.11, onde $2p < q < p \leq 0$.

As funções estruturantes estáveis são apropriadas para calcular a TD e serão estudas com detalhes no próximo capítulo.

3.3 Padrão por propagação

Algoritmo por *propagação* é um outro padrão muito estudado devido à boa eficiência dos algoritmos [VV88, Vin92, Vin93, BHJ97, FLA01, D'O01]. Como introduzida na Seção 1.2, a idéia deste padrão é processar apenas alguns pixels, inseridos em uma estrutura de dados, como *fila*, *conjunto* ou *fila hierárquica*. Com isso, o algoritmo coloca apenas os pixels que potencialmente influenciam no resultado da operação nesta estrutura. Um trabalho interessante de Vliet e Verwer [VV88] descreve o padrão por propagação para erosões, dilatações e esqueletos, porém não possui o formalismo usado em morfologia matemática.

O padrão por propagação é útil apenas em casos de operações sucessivas, onde uma função estruturante b_G pode ser decomposta como $b_G = b_1 \oplus \cdots \oplus b_k$ e $b_1 \geq \cdots \geq b_k$. Quando isto ocorre, é equivalente fazer uma operação usando b_G ou uma seqüência de operações usando os $b_i's$.

Este *padrão por propagação* possui características dos padrões paralelo e seqüencial, apresentados anteriormente. Do *padrão paralelo*, possui a característica de poder ser implementado numa arquitetura paralela para os pixels de propagação. Para o *padrão seqüencial*, em vez de possuir varreduras *raster* e *anti-raster*, o padrão por propagação é formado por uma seqüência

de padrões paralelos.

Como exemplo, é possível computar a TD usando sucessivas erosões por propagação com as funções estruturantes decompostas. Para este caso particular, após a criação da fronteira, o algoritmo entra num laço que só termina quando o mesmo estiver vazio. Dentro deste laço, um pixel por vez é retirado da fronteira para analisar seus pixels vizinhos e decidir quais destes vizinhos irão entrar numa nova fronteira. Este processo se repete até não existir nenhum conjunto fronteira. Os detalhes deste processo são apresentados no próximo capítulo.

Vincent [Vin92] abordou o problema de varredura por propagação de forma diferente, definindo os algoritmos *baseados em contorno* e subdividindo-os em duas famílias: os algoritmos *por propagação chains and loops* e os algoritmos *por propagação baseados em fila* [Vin92].

Nos algoritmos por propagação *chains and loops*, propostos em 1989 por Schmitt [Sch89], os vetores elementares que contornam a fronteira de uma imagem binária são armazenados. As transformações ocorrem usando esses vetores através de regras de propagação. Essas regras são determinadas por ângulos formados pelos pares de vetores elementares adjacentes. Por exemplo, para a grade hexagonal existem seis regras para o operador de dilatação [Vin92]. Portanto, não é necessário percorrer toda a imagem para saber quais pixels propagar, mas sim percorrer a fronteira dos objetos da imagem e verificar nas regras quais pixels são propagados.

Esse processo tem o mesmo princípio ocorrido no *padrão por propagação* usado neste documento, ou seja, percorrer apenas a fronteira dos objetos. Porém, a diferença entre estes dois padrões é que no padrão por propagação percorrem-se os pixels que contornam os objetos, colocando-os em uma estrutura de dados e entrando em um laço que analisa os vizinhos destes pixels. Estes vizinhos são determinados pela função estruturante. No padrão *chain and loop* não existe a noção de vizinhança de um pixel definida por uma função estruturante, o que dificulta ou impossibilita as transformações

genéricas ocorridas. Por exemplo, o número de regras para cada operação cresce consideravelmente com o aumento da vizinhança.

Na família de *algoritmos por propagação baseados em fila*, definida por Vincent [Vin92], os pixels de fronteira dos objetos em imagens são armazenados em uma estrutura de dados de *fila FIFO – First-In-First-Out*. Esta estrutura possui os métodos de incluir um pixel no final da fila, remover um pixel do início da fila e verificar se a fila está vazia. Esta família de algoritmos baseados em fila também foi usada por D'Ornellas [D'O01].

O uso de filas *FIFO* é uma otimização da varredura por propagação para casos particulares onde é possível trabalhar com a mesma imagem de entrada e saída e com operações idempotentes. Estes pontos serão detalhados na Subseção 4.3.3.

As maiores contribuições desta tese, no que diz respeito ao estudo de padrões de algoritmos, foram alcançadas para o *padrão por propagação*, onde foram obtidas implementações simples e eficientes de algoritmos da TD reportados na literatura. Além disso, estas implementações foram construídas através dos operadores elementares de morfologia matemática usando apenas um conjunto para armazenar a fronteira, como serão apresentadas no próximo capítulo. No final dessa seção é apresentado a condição para o uso de sucessivas erosões por funções estruturantes não crescentes.

3.3.1 Erosão por propagação

Será apresentada nesta subseção a *erosão por propagação* caracterizada por função estruturante. Não será apresentada a dilatação por propagação, pois é semelhante à erosão e foi defina por Barrera e Hirata [BHJ97].

Seja $f \in K^{\mathbb{E}}$ e $B \subseteq \mathbb{E} \oplus \mathbb{E}$ com origem. A *fronteira de propagação*, ou simplesmente *fronteira*, da erosão de f por $b \in \mathbb{Z}^B$ é o conjunto $\partial(f, b)$

definido por:

$$\partial(f,b) = \{x \in \mathbb{E} : \exists y \in B_x, f(y) > f(x)\dot{-}b(x-y)\}. \tag{3.12}$$

No Algoritmo 7 é apresentado o cômputo da fronteira de f usando uma vizinhança definida pela função estruturante b. Esta fronteira é armazenada no conjunto U, valor de saída de $\partial(f,b)$. Observe que antes de inserir um pixel no conjunto de fronteira, é feita uma verificação para que o pixel não seja inserido mais de uma vez desnecessariamente.

Algoritmo 7 Fronteira

$\partial : K^{\mathbb{E}} \times \mathbb{Z}^B \longrightarrow \mathbb{E}$
$\quad \partial(f,b) = U$

$U = \emptyset$;
$\forall x \in \mathbb{E}$
$\quad \forall y \in B_x \cap \mathbb{E}$
$\quad\quad$ **if** $f(y) > f(x)\dot{-}b(x-y)$ **e** $x \notin U$
$\quad\quad\quad U = U \cup \{x\}$;

A *erosão por propagação* de f por $b \in \mathbb{Z}^B$ é definida como:

$\forall x \in \partial(f,b),\ \forall y \in B_x \cap \mathbb{E}$,

$$\varepsilon_b^p(f)(y) = \min\{f(x)\dot{-}b(x-y)\}. \tag{3.13}$$

No Algoritmo 8 é apresentado o código que calcula a *erosão por propagação* e a nova fronteira tendo como entrada a imagem f, a função estruturante b e a fronteira U calculada por $\partial(f,b)$. Um exemplo de erosão por propagação é mostrado na Figura 3.10.

Algoritmo 8 Erosão por propagação

$\varepsilon^p : K^{\mathbb{E}} \times \mathbb{Z}^B \times \mathbb{E} \longrightarrow K^{\mathbb{E}} \times \mathbb{E}$
$\quad \varepsilon^p(f, b, U) = (g, U')$

$g = f;$
$U' = \emptyset;$
$\forall x \in U$
$\quad \forall y \in B_x \cap \mathbb{E}$
$\quad\quad$ **if** $g(y) > f(x) \dot{-} b(x - y)$
$\quad\quad\quad g(y) = f(x) \dot{-} b(x - y);$
$\quad\quad\quad$ **if** $y \notin U'$
$\quad\quad\quad\quad U' = U' \cup \{y\};$

3.3.2 Decomposição por propagação de função estruturante

De forma análoga ao ocorrido nos algoritmos paralelos e seqüenciais, a aplicação direta da erosão por uma função estruturante grande b_G é ineficiente. Contudo, é possível fazer a *decomposição por propagação da função estruturante* b_G para obter implementações rápidas, como apresentada a seguir.

É possível generalizar o Algoritmo 8 da *erosão por propagação* usando uma seqüência de erosões com funções estruturantes não crescentes, isto é, satisfazendo o critério $b_1 \geq b_2 \geq \cdots \geq b_k$, onde a relação de ordem é definida pelo reticulado $\mathbb{Z}^{B_i}$, onde $B_i \subseteq \mathbb{E} \oplus \mathbb{E}$ com origem. Esta *erosão por propagação generalizada* é mostrada no Algoritmo 9.

A condição para que o Algoritmo 9 seja válido é que a fronteira calculada na iteração i com o uso de b_i deve conter (por excesso) a fronteira associada a b_{i+1}, pois ela será usada no cálculo da erosão por b_{i+1}: $\partial(g, b_i) \geq \partial(g, b_{i+1})$.

A fronteira de f em relação à função estruturante b_i pode ser escrita

Algoritmo 9 Erosão por propagação generalizada

$\varepsilon^{pg} : K^{\mathbb{E}} \times \mathbb{Z}^{B_1} \times \mathbb{Z}^{B_2} \times \cdots \times \mathbb{Z}^{B_k} \longrightarrow K^{\mathbb{E}}$
$\quad \varepsilon^{pg}(f, b_1, b_2, \cdots, b_k) = g$

$g = f;$
$U = \partial(f, b_1);$
$\forall i \in [1, \cdots, k]$
$\quad (g, U) = \varepsilon^{p}(g, b_i, U);$

como:

$$\begin{aligned} \partial(f, b_i) &= (f \oplus \breve{b}_i) - f \quad \Longleftrightarrow \\ (f \oplus \breve{b}_i - f) &\geq (f \oplus \breve{b}_{i+1} - f) \quad \Longleftrightarrow \\ f \oplus \breve{b}_i &\geq f \oplus \breve{b}_{i+1} \quad \Longleftrightarrow \\ b_i &\geq b_{i+1}, \end{aligned}$$

que é a condição para que o Algoritmo 9 seja válido. No caso em que a erosão é idempotente este algoritmo é aplicado até a estabilidade. Quando isto ocorre, é possível obter a TD, como será visto no próximo capítulo.

Observe que a fronteira usada na erosão por propagação é formada pelos pixels que têm pelo menos um vizinho que vai ser erodido.

Analisando numa imagem binária, a fronteira é formada pelos pixels do fundo que têm pelo menos um vizinho pertencente a um objeto, como ilustrado na Figura 3.10. A fronteira usada na dilatação é formada pelos pixels do objeto que têm pelo menos um pixel de fundo. Outras operações morfológicas, como abertura e fechamento, podem ser computadas usando as erosões e dilatações por propagação, porém tomando o devido cuidado para não trocas as fronteiras. Segundo Vliet e Verwer [VV88] as rotinas para trocar uma fronteira da erosão para uma fronteira da dilatação e vice-versa são facilmente construídas.

Como sugestões para trabalhos futuros, seria interessante implementar

$$\begin{array}{ccccccccc} k & k & k & k & k & k & k & k & k \\ k & k & k & k & k & k & k & k & k \\ k & k & k & k & k & k & k & k & k \\ k & k & k & k & [0] & k & k & k & k \\ k & k & k & k & k & k & k & k & k \\ k & k & k & k & k & k & k & k & k \\ k & k & k & k & k & k & k & k & k \end{array}$$

(a) f

$$\begin{array}{ccccccccc} k & k & k & k & k & k & k & k & k \\ k & k & k & k & k & k & k & k & k \\ k & k & k & k & [1] & k & k & k & k \\ k & k & k & [1] & 0 & [1] & k & k & k \\ k & k & k & k & [1] & k & k & k & k \\ k & k & k & k & k & k & k & k & k \\ k & k & k & k & k & k & k & k & k \end{array}$$

(a) $\varepsilon_{b_4}^{p}(f)$

Figura 3.10: (a) Imagem de entrada f, onde o valor entre colchetes pertence à fronteira $\partial(f, b_4)$; (b) erosão por propagação por b_4, onde os pixels entre colchetes pertencem à nova fronteira $\partial(\varepsilon_{b_4}^{p}(f), b_4)$. b_4 é a métrica *city-block* definida no próximo capítulo.

os resultados de Vliet e Verwer [VV88] também para imagens em níveis de cinza, incluindo o trabalho da dilatação por propagação definida por Barrera e Hirata [BHJ97].

3.4 Resumo do capítulo

Nesta seção será apresentado um pequeno resumo das principais características dos estudos feito neste capítulo, no que diz respeito aos padrões de algoritmos para a erosão morfológica. A extensão para a dilatação é análoga.

A Tabela 3.1 resume as características dos padrões da erosão: as **implementações** são simples nos três padrões, como foi apresentado neste capítulo; na **arquitetura**, o padrão por propagação pode ser implementado em máquinas paralelas na erosão por uma fronteira, ou seja, é uma seqüência de algoritmos paralelos; a implementação em **hardware** é complexa no padrão por propagação, pois exige uma estrutura de **armazenamento** para a fronteira; considerando uma função estruturante constante, a **eficiência**

	paralela	**seqüencial**	**propagação**
implementação	simples	simples	simples
arquitetura	paralelo	seqüencial	paralelo
hardware	simples	simples	complexo
armazenamento	não	não	sim
eficiência	baixa	alta	alta
função estruturante	genérica	restrita	$b_1 \geq \cdots \geq b_k$
calcula a TDE	sim	não	sim

Tabela 3.1: Resumo dos padrões de algoritmos da erosão.

no caso paralelo apresenta complexidade linear com relação ao número de pixels da imagem a ser processada, porém faz acessos desnecessários (no caso seqüencial, esses acessos desnecessários são eliminados) e no padrão por propagação a complexidade é linear com relação ao número de pixels da fronteira[7]; para melhorar a eficiência dos algoritmos foi realizada a decomposição de **função estruturante** nos padrões:

paralelo: se $b_G = b_1 \oplus \cdots \oplus b_k \Longrightarrow \varepsilon_{b_G}(f) = \varepsilon_{b_k}(\cdots(\varepsilon_{b_1}(f))\cdots)$,

seqüencial: equivalência restrita entre erosão paralela e seqüencial:

- $b_G = kb^+ \vee kb^- \implies b$ genérico, porém não aplicável à TD,
- $b_G = kb^+ \oplus kb^- \implies b$ restrito, porém aplicável à TD,

propagação: se $b_G = b_1 \oplus \cdots \oplus b_k$ e $b_1 \geq \cdots \geq b_k$
$\Longrightarrow \varepsilon_{b_G}(f) = \varepsilon_{b_k}(\cdots(\varepsilon_{b_1}(f))\cdots)$;

com estas restrições, apenas o padrão seqüencial não **calcula a TDE**, pois não suporta os $b_i's$ genéricos apresentados no próximo capítulo.

[7] A complexidade de sucessivas erosões por propagação, Algoritmo 9, é mais difícil de ser analisada usando funções estruturantes variáveis. Porém em casos particulares, como para computar a TD nas métricas chanfradas definidas no próximo capítulo, é fácil ver que este algoritmo também é linear com relação ao número de pixel da imagem a ser processada, pois um pixel é inserido na fronteira no máximo uma vez.

Referências Bibliográficas

[BHJ97] J. Barrera e R. Hirata Jr. Fast algorithms to compute the elementary operators of mathematical morphology. No *Brazilian Symposium on Computer Graphics and Image Processing*, páginas 163–170, São Paulo, Brasil, Outubro 1997.

[Bor86] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34:344–371, 1986.

[Dan80] P.E. Danielsson. Euclidean distance mapping. *Computer Vision, Graphics and Image Processing*, 14:227–248, 1980.

[Dia69] R.B. Dial. Shortest-path forest with topological ordering. *Communications of the ACM*, 12(11):632–633, 1969.

[D'O01] M.C. D'Ornellas. *Algorithmic Patterns for Morphological Image Processing.* Tese de Doutorado, University of Amsterdam, Amsterdam, 2001.

[FLA01] A.X. Falcão, R.A. Lotufo e G. Araújo. The image foresting transformation. Relatório Técnico, Universidade Estadual de Campinas, 2001.

[HAS00] R.F. Hashimoto. *Mudança de Estrutura de Representação de Operadores em Morfologia Matemática.* Tese de Doutorado, Universidade de São Paulo, São Paulo - Brasil, 2000.

[LT00] N.J. Leite e M.D. Teixeira. An idempotent scale-space approach for morphological segmentation. No *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, páginas 341–350, Palo Alto, USA, Junho 2000.

[RP66] A. Rosenfeld e J.L. Pfalz. Sequential operations in digital picture processing. *Journal of the Association for Computing Machinery*, 13(4):471–494, Outubro 1966.

[Sch89] M. Schmitt. *Des algorithmes morphologiques a l'intelligence artificielle.* Tese de Doutorado, Ecole National des Mines de Paris, France, 1989.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

[SM92] F.Y.-C. Shih e O.R. Mitchell. A mathematical morphology approach to Euclidean distance transformation. *IEEE Transactions on Image Processing*, 1:197–204, 1992.

[Vin92] L. Vincent. Morphological algorithms. No E. R. Dougherty, editor, *Mathematical Morphology in Image Processing*, capítulo 8, páginas 255–288. Marcel Dekker, New York, 1992.

[Vin93] L. Vincent. Morphological grayscale reconstruction in image analysis. *IEEE Transactions on Image Processing*, 2(2):176–201, 1993.

[VV88] L.J. van Vliet e B.J.H. Verwer. A contour processing method for fast binary neighbourhood opeations. *Pattern Recognition Letters*, 7(1):27–36, 1988.

[WB88] X. Wang e G. Bertrand. An algorithm for a generalized distance transformation based on Minkowski operations. *9th ICPR*, páginas 1163–1167, 1988.

[WB92] X. Wang e G. Bertrand. Some sequential algorithms for a generalized distance transformation based on minkowski operations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1114–1121, Novembro 1992.

[WHCR95] D. Wang, V. Haese-Coat e J. Ronsin. Shape decomposition and representation using a recursive morphological operation. *Pattern Recognition*, 28(11):1783–1792, 1995.

[YTF81] S. Yokoi, Jun-I. Toriwaki e T. Fukumura. On generalized distande transformation of digitized pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(4), 1981.