Alexandre Gonçalves Silva

Uso de Árvore de Componentes para Filtragem, Segmentação e Detecção de Padrões em Imagens Digitais

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica. Área de concentração: Engenharia de Computação.

Orientador:
Prof. Dr. Roberto de Alencar Lotufo - UNICAMP

Banca Examinadora:
Prof. Dr. Clésio Luis Tozzi - UNICAMP
Prof. Dr. Francisco de Assis Zampirolli - UFABC
Prof. Dr. José Mario De Martino - UNICAMP
Prof. Dr. Roberto de Alencar Lotufo - UNICAMP
Prof. Dr. Roberto Hirata Junior - USP

Campinas, SP
Novembro de 2009

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Si28u

Silva, Alexandre Gonçalves
Uso de árvore de componentes para filtragem, segmentação e detecção de padrões em imagens digitais / Alexandre Gonçalves Silva. –Campinas, SP: [s.n.], 2009.

Orientador: Roberto de Alencar Lotufo.
Tese de Doutorado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento de imagens - Técnicas digitais. 2. Morfologia matemática. 3. Reconhecimento de padrões. 4. Árvores (Teoria dos grafos). 5. Filtros digitais. (Matemática). I. Lotufo, Roberto de Alencar. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Use of component tree for filtering, segmentation and detection of patterns in digital images
Palavras-chave em Inglês: Digital image processing, Mathematical morphology, Max-tree, Attribute filtering
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Clésio Luis Tozzi, Francisco de Assis Zampirolli, José Mario De Martino, Roberto Hirata Junior
Data da defesa: 06/11/2009
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Alexandre Gonçalves Silva

Data da Defesa: 6 de novembro de 2009

Título da Tese: "Uso de Árvore de Componentes para Filtragem, Segmentação"

Prof. Dr. Roberto de Alencar Lotufo (Presidente): _____

Prof. Dr. Francisco de Assis Zampirolli: _____

Prof. Dr. Roberto Hirata Júnior: _____

Prof. Dr. Clésio Luis Tozzi: _____

Prof. Dr. José Mario De Martino: _____

Resumo

Uma imagem em níveis de cinza pode ser interpretada como uma superfície topográfica e representada por uma árvore de componentes, baseada na relação de inclusão de regiões conexas, obtida a partir da decomposição por limiares. Medidas sobre platôs, vales ou montanhas deste relevo são úteis na caracterização de objetos de interesse em sistemas de visão computacional. Este trabalho apresenta métodos de filtragem, segmentação e reconhecimento de padrões derivados da exploração de aspectos semânticos oferecidos por essa estrutura hierárquica, construída de maneira concisa e em tempo quase-linear, mesmo com a introdução de uma série de novos atributos geométricos, topológicos e estatísticos. Havendo menos elementos a processar em relação à quantidade de *pixels* e, sendo possível a alteração da organização dos mesmos por meio de podas e enxertos, essa representação possibilita a implementação de algoritmos rápidos para operadores conexos antiextensivos. Um importante resultado da árvore estendida de atributos é a formulação genérica e determinação eficiente de novos valores de extinção, utilizados como modelo simplificado de seleção de extremos ou marcadores relevantes para reconstrução morfológica ou segmentação por regiões de influência. Propõe-se também um algoritmo unificado para pesquisa de formas conforme a análise adotada para verificação aproximada da disposição espacial de *pixels* de cada componente na árvore.

Palavras-chave: representação hierárquica, árvore de componentes, max-tree, filtragem por atributos, valores de extinção, reconhecimento de padrões, processamento baseado em regiões.

Abstract

A gray-level image can be interpreted as a topographical surface and represented by a component tree, based on the inclusion relation of connected regions, obtained by threshold decomposition. Measures on plateaus, valleys or mountains of this relief are useful in the characterization of objects of interest in computer vision systems. This work presents filtering, segmentation and pattern recognition methods from the exploration of semantic aspects provide by this hierarchical structure, whose can be constructed in a concise way and in quasi-linear time, even with the addition of a set of new geometric, topological and statistical attributes. How there is less elements to process in relation to the amount of pixels, and being able to change the organization of these through pruning and grafting, this representation allows the implementation of fast algorithms for connected anti-extensive operators. An important result of the extended attribute tree is the generic formulation and efficient determination of new extinction values, used as simplified model of selecting relevant extremes or markers for morphological reconstruction or segmentation by influence regions. A unified algorithm to search shapes is also proposed according the analysis adopted for approximate verification of the spatial layout of pixels of each component in the tree.

Keywords: hierarchical representation, component tree, max-tree, attribute filtering, extinction values, pattern recognition, region-based processing.

Agradecimentos

Sou grato ao Prof. Roberto de Alencar Lotufo pela dedicada orientação.

Aos Profs. Roberto Silvio Ubertino Rosso Jr. e Siovani Cintra Felipussi pelo incentivo.

Aos meus pais, Olenir e Elenir, e meu irmão Giuliano pelo apoio.

À Viviane pela motivação, paciência, magia e inspiração.

Ao Carlos, Franklin, Luiz, Maurício pela amizade.

Ao Rangel pela amizade e importante revisão do texto.

Ao Rubens Campos Machado pelas relevantes contribuições.

Aos demais colegas de trabalho e de pós-graduação pelas críticas e sugestões.

Ao Prof. Laurent Najman pela disponibilização de sua implementação para comparações.

À UNICAMP e UDESC pelo suporte e oportunidades.

Aos meus espirituosos pais, Olenir e Elenir

Ao meu generoso irmão Giuliano

À minha vívida Viviane

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos

$\mathcal{C}$	-	Componente conexo
$\mathcal{C}_{\mathcal{N}}$	-	Componente de nível associado a um nó $\mathcal{N}$
$\mathcal{C}_n^r(I)$	-	Componente de nível, com nível n e rótulo r, da imagem I
$\mathscr{C}_n^r(I)(p)$	-	Componente pico associado ao componente de nível $\mathcal{C}_n^r(I)$
$D_C(x_a, x_b)$	-	Dinâmica de caminho entre os pixels x_a e x_b
D_M	-	Dinâmica de um máximo regional M
$D_P(x_a, x_b)$	-	Dinâmica entre os pixels x_a e x_b
$\mathcal{D}_{\mathcal{E}}(I)$	-	Transformada de distância da imagem binária I com elemento estruturante $\mathcal{E}$
$Dil_{\mathcal{E}}(I)$	-	Dilatação da imagem I com elemento estruturante $\mathcal{E}$
$Rec_{\mathcal{E},S}(I)$	-	Reconstrução morfológica de I a partir de marcador S com elemento estruturante $\mathcal{E}$
E	-	Domínio espacial da imagem
$\mathcal{E}$	-	Elemento estruturante
$E_\mu(M)$	-	Valor de extinção, em relação a um atributo crescente μ, de um máximo regional M
$Ero_{\mathcal{E}}(I)$	-	Erosão da imagem I com elemento estruturante $\mathcal{E}$
I	-	Imagem em níveis de cinza ou binária
$I(p)$	-	Intensidade da imagem I para um pixel p
L	-	Número máximo de níveis de cinza da imagem
$\Lambda_{\mathcal{E}}(I)$	-	Rotulação da imagem binária I com elemento estruturante $\mathcal{E}$
M	-	Identificação de um máximo regional
$Maxreg(I)$	-	Máximos regionais da imagem I
$\mu(\mathcal{C}_n^r)$	-	Atributo crescente de um componente de nível $\mathcal{C}_n^r$
MT_I	-	Max-tree da imagem I
$\mathbb{N}$	-	Números naturais
$\mathbb{N}^*$	-	Números naturais, exceto o zero
$\mathcal{N}$	-	Um determinado nó da Max-tree
$Neg(I)$	-	Negação da imagem I
P_{x_a,x_b}	-	Caminho entre os pixels x_a e x_b
$\psi(I)$	-	Operador conexo aplicado à imagem I
$\psi_{\mathcal{C}^R}^{poda}$	-	Filtro de poda (antiextensivo) de uma subárvore enraizada em $\mathcal{C}^R$ de MT_I
$\psi_{\mathcal{C}^X}^{enxerto}$	-	Filtro de enxerto (antiextensivo) sobre um nó $\mathcal{C}^X$ de MT_I
$\Psi_{\mathcal{E}_d,\mathcal{E}_e}(I)$	-	Gradiente morfológico da imagem I com $\mathcal{E}_d$ de dilatação e $\mathcal{E}_e$ de erosão
$\Upsilon_{\mu,\lambda}(I)$	-	Filtro por atributo crescente μ (conexo e antiextensivo) e limiar λ
$\Upsilon_{\mu^*,\lambda}(I)$	-	Abertura por atributo crescente μ^* (conexo, antiextensivo e idempotente) e limiar λ
$\mathcal{V}_E(p)$	-	Vizinhança de um pixel p no domínio E
$X_t(I)$	-	Limiarização no nível t
$\mathbb{Z}$	-	Números inteiros

Trabalhos Publicados Pelo Autor

1. A.G. Silva, R.A. Lotufo. "New Extinction Values From Efficient Construction and Analysis of Extended Attribute Component Tree". 21^{st} *Brazilian Symposium on Computer Graphics and Image Processing* (SIBGRAPI'2008), Campo Grande, MS, p. 204-211, 2008.

2. A.G. Silva, R.A. Lotufo, R. Arthur. "Determinação Eficiente de Novos Valores de Extinção Aplicados a Problemas de Visão Computacional". $XVII$ *Congresso Brasileiro de Automática* (CBA'2008), Juiz de Fora, MG, p. 1-6, 2008.

3. A.H.S. Marcondes, M.A. Pillon, A.G. Silva. "Algoritmo de Detecção de Formas de Interesse em Imagens Digitais para uma Plataforma Distribuída". *Revista Hífen*, v. 32, n. 62, 245-252, 2008.

4. A. Körbes, A.G. Silva, R.A. Lotufo. "Attribute Sub-Tree Matching Algorithm". 8^{th} *International Symposium on Mathematical Morphology* (ISMM'2007), Rio de Janeiro, RJ, v. 2, p. 47-48, 2007.

5. A. Körbes, A.G. Silva, R.A. Lotufo. "Algoritmo para Casamento de Sub-Árvores de Atributos Aplicado ao Rastreamento de Objetos". 20^{th} *Brazilian Symposium on Computer Graphics and Image Processing - Workshop of Undergraduated Work* (SIBGRAPI-WUW'2007), Belo Horizonte, MG, p. 97-100, 2007.

6. A.G. Silva, A. Fiorese, R.E. Silva, G.B. Santos. "ANE – Árvore N-ária de Espalhamento Naturalmente Balanceada". *INFOCOMP Journal of Computer Science*, v. 6, n. 2, p. 81-90, 2007.

7. A.G. Silva, R.A. Lotufo. "Hierarchical Morphological Analysis for Generic Detection of Shapes in Grayscale Images". *Computational Modelling of Objects Represented in Images. Fundamentals, Methods and Applications* (CompIMAGE'2006), Portugal, p. 405-410, 2006.

8. A.G. Silva, S.C. Felipussi, R.A. Lotufo, G.L.F. Cassol. "k-max: Segmentation Based on Selection of Max-tree Deep Nodes". 27^{th} *Electronic Imaging (Proceedings of SPIE)* (EI'2006), v. 6064, San Jose, USA, p. 195-202, 2006.

9. A.G. Silva, R.A. Lotufo, G.L.F. Cassol. "Representação Hierárquica de Imagens para Detecção de Formas". 11^{th} *Brazilian Symposium on Multimedia and the Web* (WEBMEDIA'2005), Poços de Caldas, MG, p. 223-225, 2005.

10. A.G. Silva, R.A. Lotufo. "Detection of Lines Using Hierarchical Region Based Representation". 17^{th} *Brazilian Symposium on Computer Graphics and Image Processing* (SIBGRAPI'2004), Curitiba, PR, p. 58-64, 2004.

Capítulo 1

Introdução

Uma imagem é representada normalmente como um objeto matricial ou um conjunto de *pixels* (*picture elements*) dispostos em uma grade bidimensional. Cada pixel caracteriza uma coordenada espacial e registro associado de um ou mais valores, representando uma cor, intensidade de cinza, ou informação binária (preto ou branco). Algoritmos de processamento de imagens são, em grande parte, projetados no sentido de modificar tais valores, supondo o tratamento individual (processamento global ou ponto a ponto) ou coletivo (processamento local ou de vizinhança) destes elementos. Nesta representação, no entanto, o número de pixels a considerar é expressivo, sobretudo quando há grande quantidade de imagens ou uma sequência de vídeo a processar, mesmo com dimensões reduzidas (por exemplo, 10 imagens de tamanho 640×480 somam mais de 3 milhões de pixels). Como alternativa, pode-se imaginar a formação de regiões pelo agrupamento de pixels, a partir da definição de similaridades entre os mesmos ou regras de geração de componentes conexos baseada em limiarizações, e passar a entender a imagem como um aglomerado de regiões que, por sua vez, podem ter relacionamentos de adjacência ou hierarquia entre si.

Em se tratando de representações de imagens baseadas em regiões, destacam-se algumas estruturas em árvore [2, 3, 4, 5, 6] que vêm sendo utilizadas com sucesso em aplicações de simplificação [7, 8, 9], filtragem e segmentação [10, 5, 11, 12, 13, 14, 15, 16, 17, 18, 19], casamento, registro e reconhecimento [20, 21, 22, 23], perseguição [24, 25, 26], detecção e localização de objetos [27, 28, 29, 30], visualização volumétrica [31, 32, 33]. Estas estruturas possibilitam a exploração de aspectos semânticos [34], topológicos, estatísticos ou geométricos das regiões constituintes.

A visão de processamento sobre o domínio dessas árvores de regiões, em vez de utilização do domínio de pixels da imagem, pode gerar, em alguns casos, algoritmos mais eficientes (há, por exemplo, menos regiões que pixels a analisar) ou diferenciados (há, por exemplo, mais atributos a considerar, como área, altura, volume, entre outros além de cor). Subsídios mais consistentes para justificar estas afirmações são oferecidos ao longo deste texto.

1.1 Motivação

A quantidade e variabilidade de aplicações observadas com base em imagens representadas sob ponto de vista hierárquico são pontos iniciais relevantes e convidativos para aprofundamento de pesquisa na área. Árvores são estruturas que possibilitam representação compacta e de acesso rápido a informações. No contexto de processamento de imagens, as ligações entre seus elementos constituintes (nós) são normalmente estabelecidas no sentido de uma região conexa maior (pai) incluir uma outra menor (filho).

A Árvore de Componentes [1, 21, 5, 35, 36] é o objeto principal de investigação deste trabalho, sendo escolhida por apresentar as seguintes características: **(i)** há algoritmos em tempo quase-linear [36] para sua construção, sendo necessário definir apenas a vizinhança de pixel como parâmetro inicial; **(ii)** os componentes conexos de nível, obtidos a partir da decomposição por limiares, são exclusivos, correspondendo ao menor número de regiões conexas para a reconstrução da imagem; **(iii)** o resultado de operações sobre os nós, como poda (remoção de ramos ou subárvores completas) ou enxerto[1] (remoção de nós que não sejam folhas), são conexos antiextensivos, não criando novos contornos na imagem; **(iv)** sua organização hierárquica de regiões possibilita o descarte considerável de nós em etapas subsequentes de processamento (como casamento de imagens). **(v)** proporciona filtragem simultânea de múltiplos atributos e filtragens sucessivas no domínio da árvore, possibilitando a implementação de algoritmos eficientes, além de outras operações baseadas em topologia, geometria ou estatística.

1.2 Objetivos

A partir de algoritmos eficientes de construção da árvore de componentes, modificados para agregar novos atributos de forma incremental (no processo de inundação) e permitir medições e manipulações diferenciadas da topografia da imagem (por meio de análises estatísticas e geométricas dos componentes, e topológicas dos nós da estrutura), o presente trabalho tem como objetivos principais o melhoramento do desempenho ou extensão e generalização de operadores existentes (como, por exemplo, valores de extinção) e propostas de novos métodos de simplificação, filtragem, segmentação e detecção de formas derivados da rica informação semântica que tal estrutura dispõe.

1.3 Contribuições

A seguir, são apresentadas as principais contribuições do trabalho.

[1] Também chamado de *non-pruning strategies* [31].

- Construção da árvore: uso de vizinhança genérica de pixel e de estruturas de dados eficientes (alocações de memória em lote e tabela *hashing*) com inserção de novos atributos (grau de nó, nível do nó, número de descendentes, altura topológica, posições da semente de reconstrução da região, dos cantos superior esquerdo e inferior direito da caixa mínima envolvente ou *bounding box* do componente de nível, além de estatísticas sobre as intensidades deste como média, desvio padrão, máximo e mínimo das intensidades) em tempo de construção da árvore de componentes.

- Cálculo eficiente de novos valores de extinção (número de descendentes, altura topológica, dimensões – altura, largura e diagonal – de *bounding box*) sobre esta árvore estendida, além de um modelo simplificado de segmentação baseado em extinções relevantes.

- Desenvolvimento e aplicação, a partir de poda e enxerto seletivos de componentes, de métodos de filtragem por atributos, operadores baseados na análise topológica da árvore, e redução de constraste e limiarização baseadas nas estatísticas associadas a cada componente.

- Algoritmo genérico aproximado, sobre imagens em níveis de cinza, para detecção de linhas, formas paramétricas (retas, círculos, arcos, elipses) e *templates* binários, com aproveitamento da organização hierárquica de componentes para ganho de desempenho.

1.4 Organização da tese

A tese está organizada em 6 capítulos conforme a breve descrição que se segue: após esta introdução (Capítulo 1), tem-se definições preliminares, no Capítulo 2, importantes às formulações matemáticas posteriores e entendimento dos métodos desenvolvidos. O Capítulo 3 refere-se à representação eficiente e estendida adotada para a árvore de componentes, objeto de estudo que permeia todo o trabalho, sua comparação com outras soluções na literatura e suas aplicações. Algoritmos de filtragem e segmentação, baseados nos atributos ou na topologia da árvore, são definidos no Capítulo 4. Novos valores de extinção são propostos e suas aplicações ilustradas no Capítulo 5. Um algoritmo genérico de detecção de formas para imagens em níveis de cinza é desenvolvido no Capítulo 6. E as considerações finais, bem como descrição de trabalhos futuros, apresentadas no Capítulo 7.

Capítulo 2

Definições preliminares

O propósito deste capítulo é o de introduzir conceitos, terminologia e notações conforme sua utilização nos algoritmos dos capítulos seguintes. Definições básicas de imagem, relacionamento entre pixels[1] e entre regiões são apresentados, atributos sobre componentes conexos necessários à modelagem de novos filtros são definidos, e parte dos operadores morfológicos necessários ao entendimento do trabalho são revisados. Definições e nomenclaturas sobre as estruturas de dados *grafo*, *árvore* e *hashing* são padronizadas, de modo que a leitura do restante do texto fique clara.

2.1 Definições básicas

Definição 2.1 (Imagem em níveis de cinza) *Uma* imagem em níveis de cinza*, no contexto deste trabalho, é uma função* $I : E \rightarrow K$*, tal que* $p \in E \subset \mathbb{N}^2$*.* p *é um par ordenado ou coordenada* (p_{lin}, p_{col}) *(sequência linha e coluna) denominado* pixel*, e* $I(p)$ *consiste em intensidade luminosa* $K \in [0, L-1] \subset \mathbb{N}$*,* $L > 1$ *(usualmente* $L = 2^k$ *e* $k = 8$*) em* p*. Se* $L = 2$ *então a imagem é dita* binária*.*

Definição 2.2 (Negação) *A* negação $Neg(I)$*, de uma imagem* I *corresponde ao mapeamento inverso* $(L-1) - I$ *de intensidades. No caso de imagem binária (*$L = 2$*), também indica-se* I^C *como complemento de* I*. Formalmente:*

$$Neg(I)(p) = (L-1) - I(p), \quad \forall p \in E \tag{2.1}$$

[1] Plural de pixel – aglutinação de ***pic****ture* ***el****ement* (elemento da imagem), sendo *pix* a abreviatura de *picture* (imagem) em inglês – conforme sugestão do "Dicionário da Língua Portuguesa Contemporânea" da Academia das Ciências de Lisboa (Editora Verbo, 2001). Tais palavras são normalmente consideradas estrangeirismos pelos dicionários, mas suas traduções mais naturais, conforme regras gramaticais do português, seriam "píxel" (singular) e "píxeis" (plural).

Definição 2.3 (Vizinhança de um pixel) *A* vizinhança *de um pixel p [37] corresponde a um conjunto de coordenadas $\mathcal{V}_E(p) \subset E$ definido em relação a p no domínio E da imagem. Utiliza-se comumente $\mathcal{V}_E^{(4)}(p) = \{p + (-1,0),\ p + (0,-1),\ p + (0,1),\ p + (1,0)\} \cap E$ (*vizinhança-4*) ou $\mathcal{V}_E^{(8)}(p) = \mathcal{V}_E^{(4)}(p) \cup \{p + (-1,-1), p + (-1,1), p + (1,-1), p + (1,1)\} \cap E$ (*vizinhança-8*), tal que $d_x, d_y \in \mathbb{N}$, $p + (d_x, d_y) \in E$.*

Definição 2.4 (Caminho entre dois pixels) *O* caminho $P_{p,q}$ *de $p = p_0$ até $q = p_{n-1}$ consiste em um sequência de n pixels $P_{p_0,p_{n-1}} = (p_0, p_1, \ldots, p_{n-1})$, onde $p_i \in \mathcal{V}_E(p_{i-1})$, $\forall i \in [1, n) \subset \mathbb{N}^*$.*

Definição 2.5 (Componente conexo) *Um* componente conexo *é um conjunto maximal*[2] *de pixels $\mathcal{C} \subseteq E$, onde há sempre um caminho de p até q totalmente inserido em $\mathcal{C}$, ou $P_{p,q} \subseteq \mathcal{C}$, $\forall p, q \in \mathcal{C}$, não devendo existir $\mathcal{C}'$ tal que $\mathcal{C} \subset \mathcal{C}'$.*

Definição 2.6 (Zona plana) *Uma* zona plana *é um componente conexo $\mathcal{Z}$ da imagem I tal que $\forall p, q \in \mathcal{Z}$, $I(p) = I(q)$.*

Definição 2.7 (Histograma) *O* histograma $h_I(n)$ *corresponde ao número de vezes que o nível de cinza n ocorre na imagem I, ou, sendo $E_n = \{p \in E \mid I(p) = n\}$:*

$$h_I(n) = |E_n| = \sum_{\forall p \in E_n} 1 \tag{2.2}$$

Definição 2.8 (Limiarização) *A* limiarização $X_{t_1,t_2}(I)$ *consiste na seleção de intensidades de I em um intervalo $[t_1, t_2]$, para $t_1, t_2 \in [0, L-1]$, ou seja, $\forall p \in E$:*

$$X_{t_1,t_2}(I)(p) = \begin{cases} 1 & \textit{se} \quad t_1 \leq I(p) \leq t_2 \\ 0 & \textit{caso contrário} \end{cases} \tag{2.3}$$

Se limite superior não estiver indicado, ou $X_t(I)$, subentende-se $t_1 = t$ e $t_2 = L - 1$.

Lema 2.1 (Decomposição por limiares) *A* decomposição por limiares *consiste em se produzir L imagens binárias – cujo arranjo é, muitas vezes, denominado* conjuntos de nível *ou* level sets *– a partir de todas as limiarizações X_i possíveis ($i \in [0, L-1]$) em I. A soma de todas as imagens binárias X_i reconstrói a imagem original I:*

$$I(p) = \left(\sum_{i=0}^{(L-1)} X_i(p) \right) - 1, \quad \forall p \in E \tag{2.4}$$

Definição 2.9 (Componente de nível) *Um* componente de nível[3] $\mathcal{C}_n^r(I)$ *de uma imagem I correspon-*

[2] Um conjunto é dito *maximal* se não houver nenhum outro conjunto que o contenha.

[3] Este termo [38, 39] é também denominado *confiner* [6], *k-component* [8] ou simplesmente *(level k) component* [3].

de à imagem binária, na qual, as intensidades iguais a 1 *provêm dos pixels de um único componente conexo* r*, obtido pela limiarização* $X_n(I)$*, ou* $\forall p \in E$*:*

$$\mathcal{C}_n^r(I)(p) = \begin{cases} 1 & \textit{se } p \textit{ pertence a um componente conexo de } X_n(I)(p) \textit{ com rótulo } r \\ 0 & \textit{caso contrário} \end{cases} \tag{2.5}$$

onde $n \in [0, L-1]$ *e* $r \in [1, r_{\max}(n)]$*, sendo* $r_{\max}(n)$ *o número total de componentes conexos de* $X_n(I)$*. A Figura 2.1 exemplifica a definição (para* $n = 3$*, por exemplo,* $X_3(I)$ *gera uma imagem binária com dois componentes de nível* $\mathcal{C}_3^1(I)$*, com quatro pixels iguais a* 1*, e* $\mathcal{C}_3^2(I)$ *com um).*

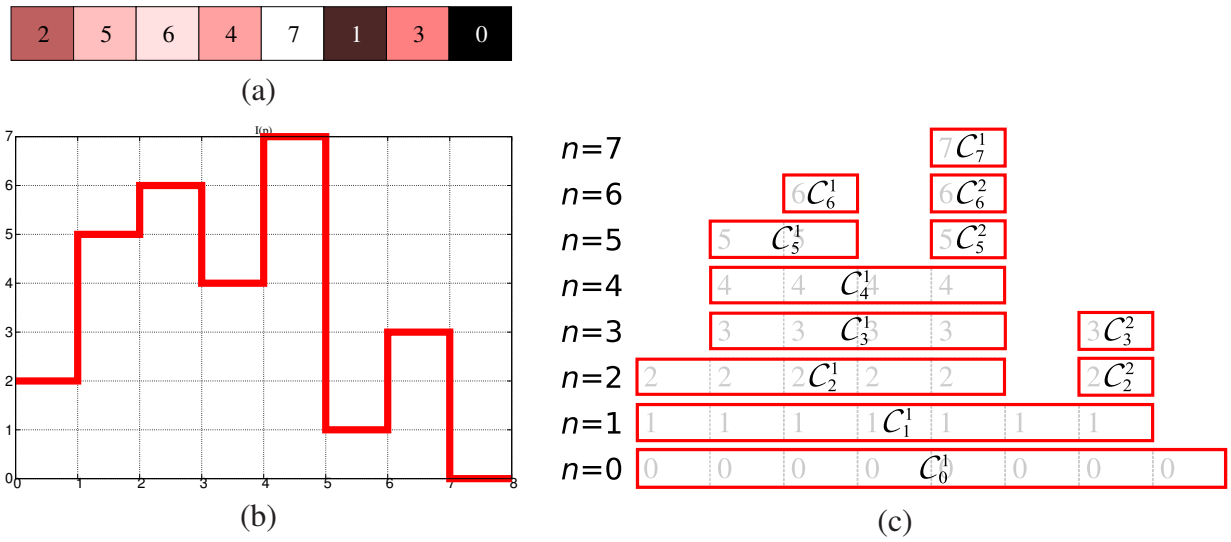


Fig. 2.1: Decomposição por limiares. (a) Pixels p e suas intensidades $I(p)$ em imagem de uma linha. (b) Perfil 1-D da função $I(p)$. (c) Componentes de nível $\mathcal{C}_n^r$, $\forall n \in [0, 7]$, $r \in [1, r_{\max}(n)]$.

Lema 2.2 (Inclusão de componentes) *A* inclusão de componentes *refere-se ao fato dos pixels iguais a* 1 *de uma limiarização em* t_1 *incluir todos o pixels iguais a* 1 *de uma limiarização em* t_2*, caso* $t_1 < t_2$*, já que* $I(p) \geq t_2$ *implica* $I(p) \geq t_1$*,* $\forall p \in E$*. Matematicamente:*

$$\{p_a \in E \mid X_{t_1}(p_a) = 1\} \supseteq \{p_b \in E \mid X_{t_2}(p_b) = 1\}, \quad \textit{se } t_1 < t_2 \tag{2.6}$$

onde $t_1, t_2 \in [0, L-1]$*. Na Figura 2.1(c), é fácil verificar a relação de inclusão entre limiarizações subsequentes.*

Definição 2.10 (Máximo regional) Máximo regional *de uma imagem* I *é uma zona plana* M *se* $I(p) > I(q)$*,* $\forall p \in M$ *e para qualquer pixel* q *vizinho de* M*. A imagem binária contendo todos os*

máximos regionais é descrita pela seguinte união de componentes de nível:

$$Maxreg(I) = \bigcup_{\forall n \in [0,L-1], \forall r} \{\mathcal{C}_n^r(I) \mid \mathcal{C}_n^r(I) \cap X_{n+1}(I) = \emptyset\} \tag{2.7}$$

Definição 2.11 (Componente pico) *Um* componente pico *ou* peak component *[40, 41] consiste na atribuição da intensidade n a todos pixels diferentes de zero de um componente de nível $\mathcal{C}_n^r(I)$, ou $\forall p \in E$, $\forall r$:*

$$\mathscr{C}_n^r(I)(p) = \begin{cases} n & \textit{se} \quad \mathcal{C}_n^r(I)(p) \neq 0 \\ 0 & \textit{caso contrário} \end{cases} \tag{2.8}$$

Definição 2.12 (Atributo crescente) *Um* atributo μ *é* crescente *se a relação de inclusão de componentes de nível implicar em relação de ordem do valor deste atributo sobre tais componentes [10]:*

$$\mathcal{C}_{n_a}^i(I) \subseteq \mathcal{C}_{n_b}^j(I) \Rightarrow \mu(\mathcal{C}_{n_a}^i(I)) \leq \mu(\mathcal{C}_{n_b}^j(I)) \tag{2.9}$$

onde $\mu(\cdot)$ é um escalar. Para ilustrar de forma didática, o volume, por exemplo, do topo de uma montanha é sempre menor que o volume da montanha como um todo. Volume, portanto, é um atributo crescente.

Definição 2.13 (Partição) *Uma* partição $P(I)$ *de uma imagem I é um conjunto de regiões distintas R_i, $i \in [1, n]$, onde a união destas forma a imagem completa, conforme descrito a seguir:*

$$P(I) = \{R_1, R_2, \ldots, R_n\} \tag{2.10}$$

onde $I = \bigcup_{i=1}^{n} R_i$, $R_i \cap R_j = \emptyset$, $\forall i, j$, $i \neq j$.

Definição 2.14 (Hierarquia de partições) *A hierarquia de partições $P_h(I)$ é um conjunto de partições, $P_1, P_2, \ldots, P_n$, de uma imagem I, em que para $\forall i, j$, $P_i = \{R_1^i, R_2^i, \ldots, R_p^i, \ldots, R_n^i\}$ são todas incluídas em regiões de $P_j = \{R_1^j, R_2^j, \ldots, R_q^j, R_m^j\}$, se $i < j$, $m < n$ e $R_p^i \subseteq R_q^j$, ou outra sub-região, se $R_p^i \cap R_q^j = \emptyset$ [42]. Uma partição de um nível k é obtida pela união de duas ou mais partições do nível $k - 1$. A Figura 2.2 exemplifica esta definição, numa sucessão de partições de um nível mais* fino *(P_1) a um mais* grosseiro *(P_5) [43].*

Definição 2.15 (Operador conexo e propriedades) *Um* operador conexo $\psi(I)$ *atua eliminando ou unindo zonas planas da imagem I, sem introduzir novos níveis de cinza (como acontece em filtros lineares) ou novas formas (como acorre em filtros morfológicos). Um operador ψ é conexo se a*

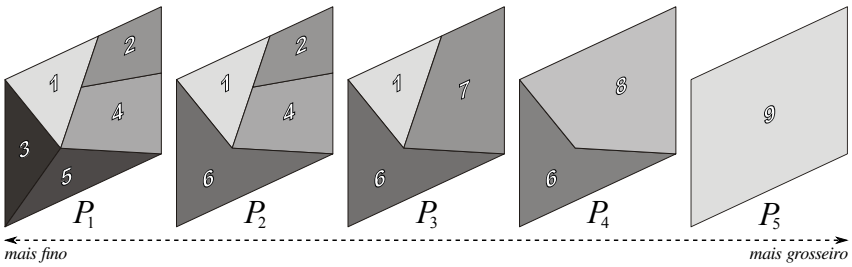


Fig. 2.2: Hierarquia de partições.

partição de zonas planas de $\psi(I)$ é menos fina que a partição de zonas planas de I [44]. Um operador conexo é:

- Antiextensivo *se satisfaz $\psi(I) \leq I$ (ou $\psi(I) \subseteq I$ para imagens binárias);*
- Crescente *se $I_a \leq I_b \Rightarrow \psi(I_a) \leq \psi(I_b)$ (ou $I_a \subset I_b \Rightarrow \psi(I_a) \subset \psi(I_b)$ para imagens binárias);*
- Idempotente *quando $\psi(\psi(I)) = \psi(I)$.*

onde $X \leq Y$ simplifica a expressão $X(p) \leq Y(p)$, $\forall p \in E$.

Definição 2.16 (Reconstrução morfológica) *A* reconstrução morfológica $Rec_{\mathcal{E},S}(I)$ *é um operador conexo antiextensivo, ou seja $Rec_{\mathcal{E},S}(I) \subseteq I$, consistindo na preservação dos componentes pico*[4]*, conexos por posições relativas $\mathcal{E}$, da imagem I com intersecção ao(s) componente(s) pico de um marcador (semente) S.*

$$Rec_{\mathcal{E},S}(I)(p) = \max_{(\forall n, \forall r)} \{\mathscr{C}_n^r(I)(p) \mid \mathcal{C}_n^r(I) \cap X_n(S) \neq \emptyset\}, \quad \forall p \in E \tag{2.11}$$

Definição 2.17 (Filtro por atributo) *Um* filtro por atributo $\Upsilon_{\mu,\lambda}(I)$ *consiste em uma reconstrução morfológica de I, na qual, cada componente pico $\mathscr{C}_n^r(I)$ em que o atributo crescente μ de seu componente de nível $\mathcal{C}_n^k(I)$ exceda um limiar λ, seja um marcador. A equação a seguir formaliza esta ideia. A Figura 2.3 mostra o resultado da filtragem de componentes para o atributo "área":*

$$\Upsilon_{\mu,\lambda}(I)(p) = \max_{(\forall n, \forall r, \forall k)} \{\mathscr{C}_n^r(I)(p) \mid \mu(\mathcal{C}_n^k) \geq \lambda\}, \quad \forall p \in E \tag{2.12}$$

[4]Outra definição tradicional de reconstrução é a de dilatações sucessivas (vide Equação 2.24 adiante) de S, por um elemento estruturante $\mathcal{E}$, condicionadas a I, até a estabilidade [45].

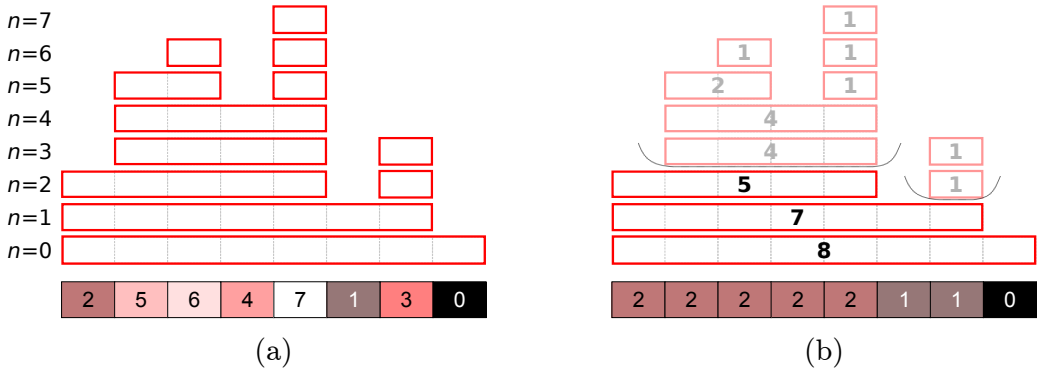


Fig. 2.3: Resultado da filtragem por atributa área. (a) Imagem original. (b) Preservação de componentes pico com $area(\mathcal{C}_n^k) \geq 5$.

Definição 2.18 (Abertura por atributo) *A* abertura por atributo *[10] é um filtro por atributo* $\Upsilon_{\mu^*,\lambda}(I)$ *idempotente, ou seja, se* $\Upsilon_{\mu^*,\lambda}\left(\Upsilon_{\mu^*,\lambda}(I)\right) = \Upsilon_{\mu^*,\lambda}(I)$*, para um atributo crescente* μ^* *que promova esta propriedade.*

2.2 Descritores de componente de nível

A determinação de descritores para um componente de nível é útil em projeto de filtros ou aberturas por atributo mencionados nas duas últimas definições. Alguns dos descritores são classificados como atributos crescentes por se ajustarem à condição da Equação 2.9, outros são estatísticos por levarem a distribuição de intensidades na imagem em consideração. A seguir, são detalhados alguns dos descritores utilizados neste trabalho.

2.2.1 Atributos crescentes

A Figura 2.4 exibe algumas medidas sobre um componente de nível. A reconstrução de todos os componentes picos, cujos componentes de nível estão incluídos neste, formam um relevo, do qual pode-se extrair altura, área, volume, altura e largura da mínima caixa envolvente definidos a seguir.

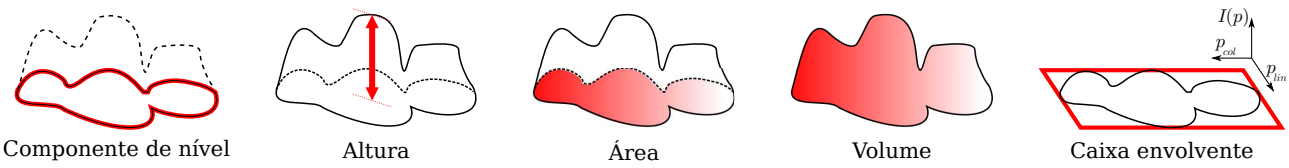


Fig. 2.4: Atributos crescentes para o topo do relevo de uma região da imagem.

Definição 2.19 (Altura) *A* altura μ_{altura} *de* $\mathcal{C}_n^r(I)$ *corresponde à diferença entre o maior nível de cinza (*$\max(n_d)$*) de componentes incluídos em* $\mathcal{C}_n^r(I)$ *e o nível de cinza* n*:*

$$\mu_{altura}(\mathcal{C}_n^r(I)) = \max_{\forall n_d > n, \forall k}(n_d) - n, \quad \mathcal{C}_{n_d}^k(I) \subset \mathcal{C}_n^r(I) \tag{2.13}$$

Definição 2.20 (Área) *A* área μ_{area} *refere-se ao número de pixels diferentes de zero de (pertencentes a)* $\mathcal{C}_n^r(I)$*, ou, sendo* $\mathcal{C}_n^r(I)$ *uma imagem binária:*

$$\mu_{area}(\mathcal{C}_n^r(I)) = \sum_{\forall p \in E} \mathcal{C}_n^r(I)(p) \tag{2.14}$$

Definição 2.21 (Volume) *O* volume μ_{volume} *é igual a soma das áreas de todos os componentes incluídos em* $\mathcal{C}_n^r(I)$*:*

$$\mu_{volume}(\mathcal{C}_n^r(I)) = \sum_{\forall n_d > n, \forall k} \mu_{area}(\mathcal{C}_{n_d}^k(I)), \quad \mathcal{C}_{n_d}(I) \subset \mathcal{C}_n^r(I) \tag{2.15}$$

Definição 2.22 (Altura e largura da caixa envolvente) *A* altura μ_{abbox} *e largura* μ_{lbbox} *da* caixa envolvente *ou* bounding box[5] *de um componente de nível* $\mathcal{C}_n^r(I)$*, são, respectivamente, as diferenças entre a maior e menor linha, e entre a maior e menor coluna,* $\forall p \in E$ *tal que* $\mathcal{C}_n^r(I)(p) \neq 0$*:*

$$\mu_{abbox}(\mathcal{C}_n^r(I)) = \max_{\forall p \in \mathcal{C}_n^r(I)}(p_{lin}) - \min_{\forall p \in \mathcal{C}_n^r(I)}(p_{lin}) \tag{2.16}$$

$$\mu_{lbbox}(\mathcal{C}_n^r(I)) = \max_{\forall p \in \mathcal{C}_n^r(I)}(p_{col}) - \min_{\forall p \in \mathcal{C}_n^r(I)}(p_{col}) \tag{2.17}$$

2.2.2 Atributos estatísticos

Definição 2.23 (Média) *A* média μ_μ *de* $\mathcal{C}_n^r(I)$ *é a soma dos níveis de cinza de* I*, para os pixels pertencentes ao componente, dividida pela quantidade destes pixels (ou área):*

$$\mu_\mu(\mathcal{C}_n^r(I)) = \frac{\sum_{\forall p \in \mathcal{C}_n^r(I)} I(p)}{\mu_{area}(\mathcal{C}_n^r(I))} \tag{2.18}$$

Definição 2.24 (Desvio padrão) *O* desvio padrão μ_σ *de* $\mathcal{C}_n^r(I)$ *é a medida de dispersão da média dos níveis de cinza de* I *para os pixels deste componente:*

$$\mu_\sigma(\mathcal{C}_n^r(I)) = \left(\frac{\sum_{\forall p \in \mathcal{C}_n^r(I)} [I(p) - \mu_\mu(\mathcal{C}_n^r(I))]^2}{\mu_{area}(\mathcal{C}_n^r(I)) - 1} \right)^{\frac{1}{2}} \tag{2.19}$$

[5] A caixa mínima envolvente ou *minimum bounding box*, de forma mais precisa, possui lados perpendiculares entre si e paralelos aos eixos de coordenadas da imagem.

para $\mu_{area}(\mathcal{C}_n^r(I)) > 1$*; caso contrário,* $\mu_\sigma(\mathcal{C}_n^r(I)) = 0$*.*

Definição 2.25 (Mínimo e máximo) *Seguem a* mínima μ_{min} *e* máxima μ_{max} *intensidades* I *no domínio do componente* $\mathcal{C}_n^r(I)$*:*

$$\mu_{min}(\mathcal{C}_n^r(I)) = \min_{\forall p \in \mathcal{C}_n^r(I)} I(p) \tag{2.20}$$

$$\mu_{max}(\mathcal{C}_n^r(I)) = \max_{\forall p \in \mathcal{C}_n^r(I)} I(p) \tag{2.21}$$

2.2.3 Outros descritores

Definição 2.26 (Perímetro) *O* perímetro *corresponde ao tamanho do contorno, ou à quantidade de pixels* p *de* $\mathcal{C}_n^r(I)$ *com intensidades iguais a* 1 *que tenham ao menos um vizinho* q *com intensidade* 0*.*

$$p(\mathcal{C}_n^r(I)) = \sum_{\forall p \in \mathcal{C}_n^r(I), \forall q \notin \mathcal{C}_n^r(I)} \{\mathcal{C}_n^r(I)(p) \mid p \in \mathcal{V}_E^{(4)}(q)\} \tag{2.22}$$

Definição 2.27 (Centroide) *O* centroide *de* $\mathcal{C}_n^r(I)$ *é uma única coordenada* $c = (c_{lin}, c_{col}) \in E$*, onde* c_{lin} *é a média das linhas e* c_{col}*, a média das colunas,* $\forall q \in E$*, tal que* $\mathcal{C}_n^r(I)(q) \neq 0$*. De outra forma, sendo* $E' = \{q \in E \mid \mathcal{C}_n^r(I)(q) \neq 0\}$ *e* $n = \mu_{area}(\mathcal{C}_n^r(I))$*:*

$$c(\mathcal{C}_n^r(I)) = \frac{1}{n} \sum_{\forall p \in E'} p \tag{2.23}$$

2.3 Morfologia matemática

Os algoritmos de filtragem e segmentação apresentados ao longo do texto requerem o conhecimento de alguns operadores morfológicos [38] apresentados nesta seção. Os mesmos são, em geral, descritos para imagens em níveis de cinza. Quando seu uso for restrito a imagens binárias, a definição explicita esta condição. São considerados apenas elementos estruturantes planares (a Def. 2.28 apresentada a seguir não abrange a representação de funções estruturantes), suficientes para os propósitos deste trabalho.

Definição 2.28 (Elemento estruturante) *O* elemento estruturante *(planar)* $\mathcal{E}$ *está associado a um conjunto de posições relativas de um pixel apropriado à análise morfológica. Algumas formas usuais:* $\mathcal{E}_{cruz} = \{(-1,0),(0,-1),(0,0),(0,1),(1,0)\}$*;* $\mathcal{E}_{caixa} = \mathcal{E}_{cruz} \cup \{(-1,-1),(-1,1),(1,-1),(1,1)\}$*.*

Definição 2.29 (Dilatação) *A* dilatação *[46, 47, 48, 38] é um filtro não-linear* $Dil_\mathcal{E}(I)$*, ou* $I \oplus \mathcal{E}$*, que transforma a intensidade de cada pixel* p *pelo máximo entre as intensidades dos pixels relativos a* p

definidas por $\mathcal{E}$, com propósito de aumento das regiões claras da imagem I, conforme:

$$Dil_{\mathcal{E}}(I)(p) = \max_{\forall e \in \mathcal{E}}\{I(p-e)\}, \quad \forall (p-e) \in E \tag{2.24}$$

Definição 2.30 (Erosão) *A* erosão *[46, 47, 48, 38] $Ero_{\mathcal{E}}(I)$, ou $I \ominus \mathcal{E}$, é operador dual da dilatação, ou matematicamente, $Neg(Dil_{\mathcal{E}}(Neg(I)))$, e transforma a intensidade de cada pixel p pelo mínimo das intensidades dos pixels relativos a p definidas por $\mathcal{E}$, com propósito de aumento das regiões escuras da imagem I (e consequente redução das áreas claras), conforme:*

$$Ero_{\mathcal{E}}(I)(p) = \min_{\forall e \in \mathcal{E}}\{I(p+e)\}, \quad \forall (p+e) \in E \tag{2.25}$$

Definição 2.31 (Gradiente morfológico) *O* gradiente morfológico $\Psi_{\mathcal{E}_d,\mathcal{E}_e}(I)$ *[38] consiste em uma erosão (com $\mathcal{E}_e$) subtraída de uma dilatação (com $\mathcal{E}_d$) e determina o realce (imagens em níveis de cinza) ou detecção (imagens binárias) de contornos:*

$$\Psi_{\mathcal{E}_d,\mathcal{E}_e}(I) = Dil_{\mathcal{E}_d}(I) - Ero_{\mathcal{E}_e}(I) \tag{2.26}$$

Definição 2.32 (Transformada de distância) *A* transformada de distância $\mathcal{D}_{\mathcal{E}}(I)$ *de uma imagem binária I consiste na atribuição, aos pixels diferentes de zero, de distância ao pixel igual a zero mais próximo [49]. Consiste em uma soma de n erosões sucessivas da imagem I até a estabilidade:*

$$\mathcal{D}_{\mathcal{E}}(I)(p) = I(p) + \sum_{i=1}^{\infty} I_i(p), \quad \textit{tal que } I_i = Ero_{\mathcal{E}}(I_{i-1}), \quad \forall p \in E \tag{2.27}$$

onde $I_0 = I$.

Definição 2.33 (Rotulação) *A* rotulação $\Lambda_{\mathcal{E}}(I)$ *de uma imagem binária I consiste na atribuição de um rótulo $r \in \mathbb{N}^*$ a $I(p_r)$, $p_r \in \mathcal{C}_r$, para cada um de seus componentes conexos ($\mathcal{C}_1$, $\mathcal{C}_2$, $\cdots$) [50]. Segue formulação a partir de uma soma de reconstruções (vide Def. 2.16) $\forall p \in E$:*

$$\Lambda_{\mathcal{E}}(I)(p) = \sum_{r=1}^{\infty} \left(r \cdot Rec_{\mathcal{E},S_r}(I)\right)(p) \tag{2.28}$$

sendo S_r a semente de um componente conexo não reconstruído até o termo anterior $r-1$ do somatório:

$$S_r(q) = \begin{cases} 1 & \textit{para um único pixel } q \in E, \textit{ tal que } I(q) \neq 0 \textit{ e } \sum_{k=1}^{k=r-1} \left(k \cdot Rec_{\mathcal{E},S_k}(I)\right)(q) = 0 \\ 0 & \textit{para os demais pixels} \end{cases}$$

2.4 Estruturas de dados

Seguem as definições e nomenclaturas associadas às representações em grafos, árvores e *hashing*, importantes ao trabalho.

Grafo

Grafo $G(V, A)$ é uma estrutura de dados para modelagem de diversos problemas computacionais, incluindo representação de imagens. É definido por: V – conjunto não vazio de *vértices* ou *nós* do grafo; e A – conjunto de pares (v, w) de vértices distintos, $v, w \in V$, denominado *arcos* (para digrafos) ou *arestas*. Um arco ou aresta (v, w) determina que v e w são nós *adjacentes*. Um grafo é *orientado* (digrafo) se o par (v, w) é ordenado, ou seja, se a relação entre v e w não é simétrica (por exemplo, relação de amizade é simétrica; relação pai(mãe)-filho(a) é assimétrica). Se orientado, a adjacência entre os vértices se especializa em *sucessor* e *antecessor* (se há um arco de v para w, v é antecessor e w é sucessor). A *ordem* de um grafo é indicada por $|V|$ e consiste na cardinalidade do conjunto V ou quantidade de seus nós. O *tamanho* do grafo, $|A|$, é a quantidade de arestas. O *grau* de um nó $v \in V$ é igual ao número n de outros nós $w_1, w_2, \ldots, w_n \in V$ conectados a v (ou seja, $\{(v, w_1), (v, w_2), \ldots, (v, w_n)\} \subseteq A$). A cada aresta, pode ser atribuído um peso $\mathrm{w}_{v,w}$. Neste caso, cada elemento de A passa a ser $(v, w, \mathrm{w}_{v,w})$ e o grafo, ter a denominação *valorado*, como ilustrado na Figura 2.5. Um *laço* é uma aresta do tipo (v, v), ou relação de um nó a si próprio (vide penúltimo arco de A no $G(V, A)$ da Figura 2.5). Um *caminho* de v_1 a v_n é uma lista de nós $(v_1, v_2, \ldots, v_n)$, tal que sejam v_i e v_{i+1} sejam adjacentes, $\forall i \in [1, n-1]$. O caminho é simples se não inclui duas vezes a mesma aresta. Um *ciclo* é uma caminho simples e fechado (v_1 e v_n é mesmo nó). Se v_i é antecessor de v_{i+1}, $\forall i \in [1, n-1]$ no caminho, então tem-se um *caminho orientado* (neste caso, ciclo recebe a denominação especializada de *circuito*). O *comprimento do caminho* entre nós v e w é igual ao número de arcos do caminho entre v e w. Um grafo é *conexo* se, $\forall v, w \in V$, há um caminho de v a w. O *custo do caminho* é dado por uma função dos pesos das arestas no $C(v_1, v_2, \ldots, v_n)$. O custo entre dois nós $C^*(v, w)$ é dado pelo menor custo entre todas os caminhos possíveis entre v e w.

Árvore

Uma *árvore* é um grafo $T(V, A)$ conexo e acíclico (sem ciclo) apropriado para a representação de problemas computacionais em que os elementos tratados podem ser organizados de forma hierárquica. Algoritmos procuram se aproveitar do princípio da decomposição inerente a este modelo, algo semelhante à *divisão e conquista* [51]. No contexto deste trabalho, apenas árvores com raiz e orientadas serão tratadas. Neste sentido, seguem definições de terminologia associada: nó v é *ancestral* de w se há um caminho orientado de v a w (nós distintos) e, neste caso, w é *descendente* de v; um

$$
\begin{aligned}
V &= \{\ 1,\ 2,\ 3,\ 4,\ 5\ \} \\
A &= \{\ (1,2,\mathrm{w}_{1,2}), \\
&\quad (2,3,\mathrm{w}_{2,3}),\ (2,4,\mathrm{w}_{2,4}),\ (2,5,\mathrm{w}_{2,5}), \\
&\quad (3,4,\mathrm{w}_{3,4}),\ (3,5,\mathrm{w}_{3,5}), \\
&\quad (4,3,\mathrm{w}_{4,3}),\ (4,4,\mathrm{w}_{4,4}), \\
&\quad (5,1,\mathrm{w}_{5,1})\ \}
\end{aligned}
$$

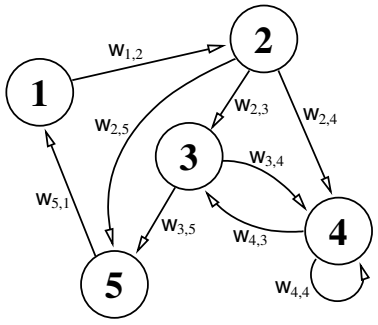


Fig. 2.5: Exemplo de definição de nós V e arcos A de digrafo valorado $G(V, A)$ (à esquerda) e sua representação gráfica (à direita).

nó v_r é dito *raiz* se for ancestral de todos os outros nós ou, em outras palavras, se não tiver nenhum ancestral ou arco entrante (todos os nós, exceto a raiz, devem ter exatamente um arco entrante); um nó v é pai(mãe) de um nó w se houver um arco orientado no sentido de v para w e, neste caso, w é filho(a) de v (nós adjacentes); um nó v é *irmão(ã)* de w se v e w têm o mesmo pai; há sempre um único caminho entre dois nós, e há um único caminho orientado de v para w se, e somente se, v é ancestral de w (ou w descendente de v). Um nó v qualquer e todos os seus descendentes formam um subgrafo especial denominado *subárvore* (se v é raiz então a subárvore é igual a árvore). O nó v_f que não apresenta descendente é nomeado de *folha*. Percebe-se a recursividade na definição de uma árvore como composição de subárvores. O *nível topológico* de um nó v é igual ao comprimento do caminho entre a raiz v_r e v (a raiz está no nível 0, os filhos da raiz, no 1, e assim por diante). Se todos os níveis topológicos estiverem com a quantidade máxima de nós, a árvore é dita *completa*. Não havendo preenchimento total apenas no último nível, então denomina-se *quase-completa*. A *altura topológica* de uma árvore é igual ao comprimento máximo entre todos os caminhos orientados possíveis (ou número de arcos da raiz a uma folha com maior nível topológico). Também são comumente utilizados *ordem* da árvore e *grau* do nó, conforme a definição já feita para grafos.

Hashing

Um *hashing* refere-se basicamente[6] a uma tabela H, na qual, por meio de uma *função de hashing* f_h, uma posição (*bucket*) k é mapeada a partir de uma chave de pesquisa ch, ou seja, a posição $k = f_h(ch)$ contém a informação desejada $H(k)$. O objetivo, portanto, é a obtenção de uma informação, indexada por uma chave, com um número mínimo de comparações. Se H tem N posições, pode-se utilizar, por exemplo, $f_h(ch) = ch \bmod N$, sendo *mod* o operador de resto da divisão inteira. Nota-se que, para a função de espalhamento implementada com *mod*, pode ocorrer de duas chaves distintas

[6]Há inúmeras variações de *hashing* se modeladas com base no problema a ser resolvido.

serem mapeadas na mesma posição. Esta situação é denominada *colisão* e um método de *resolução de colisões* se faz necessário (várias chaves mapeadas em uma mesma posição podem se organizar em uma lista, por exemplo). Idealmente não há colisão se a tabela tiver um tamanho adequado e uma função de espalhamento perfeita [52]. Uma estrutura de dados híbrida, a partir de um modelo hierárquico de tabelas *hashing* [53] é uma opção eficiente de implementação no sentido de reduzir o número de comparações necessárias à obtenção de uma informação e, desta forma, evitar colisões. Espera-se, no entanto, que haja um compromisso entre eficiência e uso de memória, conforme o conjunto de dados tratado e recursos de máquina que se dispõe.

2.5 Considerações sobre o capítulo

Neste capítulo, foi apresentado o ferramental matemático e computacional básico ao desenvolvimento do trabalho. A definição de componente de nível, relativo a um agrupamento de pixels (componente conexo) obtido na decomposição por limiares da imagem, assim como o histograma e estruturas de dados ilustradas (grafo, árvore e *hashing*), são essenciais para a implementação, de forma eficiente, da árvore de componentes (conforme será visto, em detalhes, no próximo capítulo). Todas as definições seguem padronização de sintaxe em função desta representação (por exemplo, o atributo área só diz respeito a quantidade de pixels pertencentes a um único componente de nível). Os atributos mencionados, exceto o perímetro, são determinados em tempo de construção da árvore, e estão associados a cada componente de nível da imagem, sendo importantes na elaboração, por exemplo, de filtros por atributos (Cap. 4). Os atributos crescentes, em especial, auxiliam no cálculo de valores de extinção para determinação de máximos regionais relevantes por algum critério (área, altura, volume, entre vários outros) e simplificação ou segmentação da imagem (Cap. 5). Atributos estatísticos são úteis em estratégias de limiarização ou binarização adaptativa da imagem. Outros atributos, como o centroide, podem ser usados na caracterização de forma de cada componente (Cap. 6).

Capítulo 3

Árvore de componentes

Estruturas hierárquicas vêm sendo utilizadas, de diversas maneiras, para representação e operações de imagens. Variações importantes do tema, constantemente citadas na literatura, são descritas a seguir. Após esta breve exposição, a árvore de componentes passa a ser detalhada, servindo de elo de ligação para todos os algoritmos propostos no trabalho. Eis a revisão de algumas árvores para processamento de imagens:

- **Quadtree.** No contexto de tratamento de imagens, a *quadtree* [54] corresponde a uma representação de regiões [55], na qual, sucessivas divisões da imagem são feitas em quatro partes iguais. Se todas as intensidades (ou cores) de um quadrante não satisfaz um critério de similaridade, então nova subdivisão em subquadrantes é realizada. Caso satisfaça, o algoritmo armazena a região não a subdividindo mais. A similaridade pode se definida, por exemplo, a partir da diferença entre máxima e mínima intensidade dos pixels de um quadrante, ou pela medição do padrão de variação das intensidades daquele bloco de pixels [56]. Portanto, *quadtree* pode ser interpretada como uma técnica de representação da imagem em diferentes resoluções e, neste sentido, compressão é uma aplicação imediata comum desta ferramenta.

- **Árvore dos Lagos Críticos.** Pode-se pensar a imagem como uma superfície topográfica e em lagos formados por diques como *linhas divisores de água* [57] provenientes da elevação do nível de água a partir de fontes localizadas em todos os mínimos regionais (*watershed clássico*). Supondo que um processo contínuo de inundação, a partir dos mínimos regionais, se inicie (regiões da imagem ou folhas da árvore), em certo momento, as águas de dois lagos se unem (fusão de duas regiões ou dois nós se ligando a um nó pai na árvore). E este processo de união se repete até que se tenha apenas um lago (região única com todos os pixels da imagem ou raiz da árvore). Tem-se construído então a estrutura denominada *árvore dos lagos críticos* [2]. Não só a profundidade, mas a área ou volume dos lagos pode determinar a ordem de fusão [38]. Sua

utilização é especialmente voltada para segmentação de imagens [14].

- **Árvore de Partição Binária.** A *árvore de partição binária* [58, 59] é uma generalização das árvores para representação de multirresolução de imagens. O conjunto de regiões e a forma que suas intercalações acontecem, duas a duas, são bastante flexíveis. A partição pode ser definida inicialmente por todas as zonas planas [60] ou qualquer outra aproximação [61, 62] como *watershed*[1] ou método de crescimento de regiões a partir de sementes pré-definidas. Suas aplicações envolvem desde métodos de filtragem [4] a análise de vídeo [63].

- **Árvore de Componentes.** A *árvore de componentes* [5, 35, 36] consiste em uma estrutura hierárquica de regiões baseada no relevo formado pela decomposição por limiares de uma imagem em níveis de cinza. Outras denominações são encontradas na literatura para esta mesma (ou similar) representação: dendrograma [64], árvore de conectividade [65], árvore de confinamento [21, 6] ou Max-tree[2] [1, 11, 32]. Os dois termos quase que exclusivamente utilizados atualmente são árvore de componentes e Max-tree[3]. Neste trabalho, os dois nomes são encarados como sinônimos, sendo o último adotado preferencialmente, sobretudo nos algoritmos desenvolvidos, onde o símbolo MT_I indica a Max-tree de uma imagem I.

- **Outras árvores.** Outras estruturas hierárquicas voltadas ao processamento de imagens podem ser citadas: *árvore escala* [66, 67], baseada em decomposição da imagem por granularidade de suas regiões constituintes, útil em filtragem e segmentação [68]; *árvore de inclusão* [69], onde os componentes conexos são definidos por linhas de nível (e não por decomposição por limiares), e a relação de inclusão geométrica destas definem a estrutura hierárquica, utilizada em simplificação e comparação de imagens; *árvore de formas* [70], na qual, é feita uma hierarquia de contornos pela decomposição de uma curva em seu ponto médio, para classificação e detecção de objetos baseadas no casamento de subcurvas. Nesta linha, também há a *árvore de singularidades* de curvas [71] para casamento de formas binárias. Pode-se citar ainda árvores provenientes do particionamento de grafo como modelo do relacionamento de pixels da imagem, como *árvores direcionadas* [72] ou *transformada imagem-floresta* [73], úteis em segmentação e operações morfológicas.

[1] Ao considerar, por exemplo, a profundidade do *watershed* como critério de junção de regiões, a árvore de partição binária se especializa na árvore dos lagos críticos.

[2] Max-tree refere-se ao algoritmo de [1] para implementação eficiente da árvore de componentes.

[3] Alguns autores [5, 35, 23] preferem diferenciar as duas estruturas, definindo árvore de componentes como sendo a relação de todos os componentes de nível possíveis, e Max-tree como sendo formada, de maneira mais compacta, apenas pelos componentes com zonas planas na imagem.

3.1 Definição

Antes de descrever a construção da Max-tree adotada, convém apresentar a Figura 3.1 para ilustração da estrutura de uma imagem exemplo de altura 4 e largura 8 (Fig. 3.1a). O relevo é definido assumindo uma altitude dada pelo nível de cinza de cada pixel (Fig. 3.1b). A raiz da árvore $\mathcal{N}^R$ corresponde à região $\mathcal{C}_{\mathcal{N}^R}$ formada pelos pixels cujas intensidades são iguais ou maiores à menor intensidade, no caso 0, de I (ou seja, componente de nível, cujos pixels correspondem a todo domínio E da imagem). Internos a esta região, observam-se, supondo uso de vizinhança-8, três componentes de nível formados por pixels com intensidades maiores ou iguais a 50, resultando nos três filhos da raiz (observa-se que um destes três componentes tem 100 como menor intensidade, sendo este ligado diretamente como filho da raiz). A próxima limiarização, supondo apenas intensidades presentes em zonas planas, é de níveis maiores ou iguais a 100, gerando novos três componentes, filhos dos componentes anteriores que os contêm e, assim, sucessivamente (Fig. 3.1c), até que se tenham as folhas da árvore correspondentes aos máximos regionais de I. A formação estrutural da superfície topográfica da imagem e relações de inclusão entre os componentes de nível determinam a topologia de nós da árvore (Fig. 3.1d).

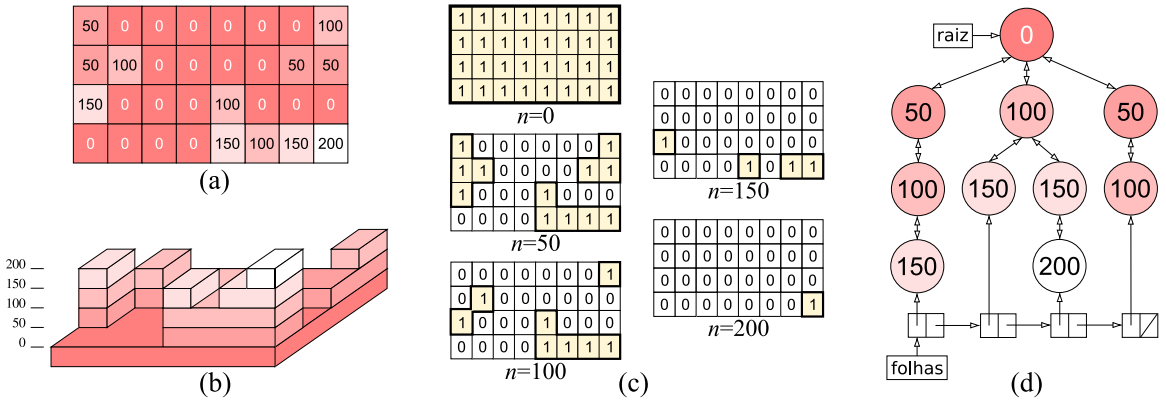


Fig. 3.1: Exemplo de construção da Max-tree. (a) Imagem I. (b) Superfície topográfica de I. (c) Componentes de nível n. (d) Max-tree MT_I com vizinhança-8.

O isolamento de um componente de nível, a partir de um nó da árvore, é uma operação auxiliar desejável em reconstruções após uma filtragem, por exemplo. A Figura 3.2 ajuda a entender o procedimento para se mapear um nó $\mathcal{N}_{\mathcal{C}_n^r}$, dado $n = 50$, em seu componente de nível $\mathcal{C}_n^r$, considerando ainda a imagem I da Figura 3.1. Os passos geram as seguintes imagens: **(i)** R a partir da atribuição de 1 a todos os pixels maiores ou iguais a n (Fig. 3.2a com $n = 50$); **(ii)** S a partir da atribuição de 1 somente aos pixels iguais a n (Fig. 3.2b com $n = 50$); **(iii)** reconstrução das regiões R com os marcadores S. Se a reconstrução resultar em k componentes de nível diferentes, então há k nós com o mesmo valor n (Fig. 3.2c). A seleção da correta associação entre nó, com rótulo r, e sua região,

neste caso, é baseada na matriz de rótulos *status* gerada na construção da Max-tree [74, 1, 11], cujo algoritmo será detalhado na Seção 3.4 (Alg. 3.1).

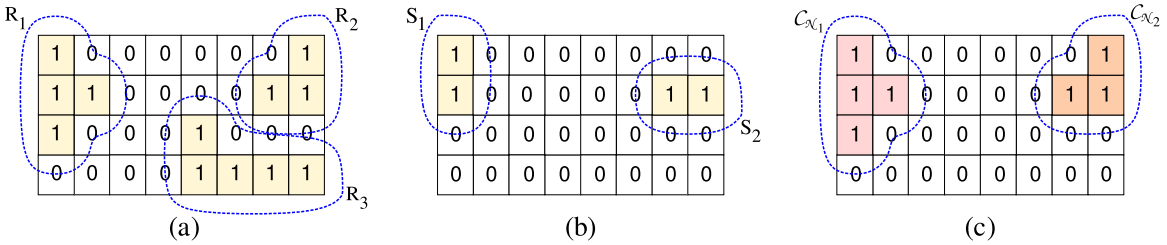


Fig. 3.2: Exemplo de reconstrução de nós $\mathcal{N}$, com valor 50, em seus componentes de nível $\mathcal{C}_{\mathcal{N}}$ da imagem I da Figura 3.1. (a) Limiarização $R = X_n$ para $n = 50$. (b) Marcadores S para nível $n = 50$. (c) Reconstrução $Rec_{\mathcal{E}_{caixa},S}(R)$.

A ideia deste trabalho é a de representar uma imagem I em uma estrutura Max-tree MT_I eficientemente, onde cada nó $\mathcal{N}$ corresponde a uma imagem binária $\mathcal{C}_{\mathcal{N}}$ (de mesmo tamanho de I) contendo um único componente de nível, de modo que algoritmos diferenciados de processamento de imagens possam ser desenvolvidos. De maneira formal, define-se Max-tree como:

Definição 3.1 (Max-tree) *A* Max-tree MT_I *é uma estrutura hierárquica formada a partir do relacionamento de inclusão entre componentes de nível* $\mathcal{C}_n^r$ *(limiar* n *e rótulo* r*). Um nó* $\mathcal{N}_{\mathcal{C}_{n_a}^i}$ *é ancestral de* $\mathcal{N}_{\mathcal{C}_{n_b}^j}$ *se, e somente se,* $\mathcal{C}_{n_a}^i$ *contém* $\mathcal{C}_{n_b}^j$*, onde* $n_a < n_b$ *[11].* $\mathcal{N}^R$ *é raiz se* $\mathcal{C}_{\mathcal{N}^R} = \mathcal{C}^1_{\min(I)}$*, onde* $\min(I)$ *é a menor intensidade da imagem.* $\mathcal{N}^L$ *é um nó folha se* $\mathcal{C}_{\mathcal{N}^L}$ *é um máximo regional ou* $\mathcal{C}_{\forall\mathcal{N}\neq\mathcal{N}^L} \nsubseteq \mathcal{C}_{\mathcal{N}^L}$*. Se* $\mathcal{C}_{n_a}^i = \mathcal{C}_{n_b}^j$*, sendo* $n_a < n_b$*, um nó* $\mathcal{N}_{\mathcal{C}_{n_b}^j}$ *é definido somente para* $\mathcal{C}_{n_b}^j$*, pois* $\mathcal{C}_{n_a}^i(p) < I(p), \forall p \in \mathcal{C}_{n_a}^i$ *(apenas* $\mathcal{C}_{n_b}^j$ *contém pixel em zona plana).*

De forma análoga e oposta, define-se Min-tree:

Definição 3.2 (Min-tree) *A* Min-tree mt_I *corresponde à Max-tree da negação da imagem, ou* $mt_I = MT_{Neg(I)}$*, de modo que,* $L-1$ *(máxima intensidade representável) subtraído da máxima intensidade de* I *caracterize a raiz, e os mínimos regionais de* I *passem a ser as folhas da Min-tree.*

Segundo a definição de Max-tree, cada nó corresponde a um componente de nível que contém pixels em zonas planas (vide Sec. 2.1) da imagem [1, 36]. Alguns autores [35, 23] preferem, no entanto, considerar que esta ideia se aplica apenas à Max-tree, sendo a Árvore de Componentes composta por todos os limiares possíveis, independente de conter ou não um pixel em uma zona plana. Em outras palavras, independente de haver componentes de nível idênticos (mesmo domínio de pixels) obtidos em limiarizações subsequentes. A Figura 3.3 mostra esta diferenciação. No contexto deste trabalho, o conjunto de nós, dessa figura, correspondente à Max-tree é também a única configuração possível da Árvore de Componentes. Os termos, portanto, são encarados como sinônimos neste texto.

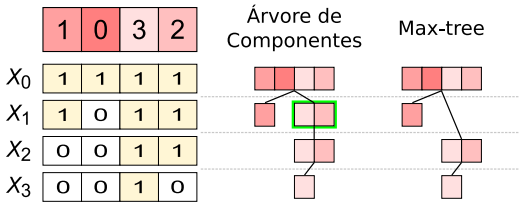


Fig. 3.3: Imagem e sua decomposição por limiares (esquerda) e a diferenciação entre Árvore de Componentes (centro) e Max-tree (direita) adotada por alguns autores (no presente trabalho, a estrutura mais concisa à direita é a única possível, sendo também denominada Árvore de Componentes).

3.2 Aplicações

A representação hierárquica de componentes de nível de uma imagem vem sendo utilizada como estrutura de dados auxiliar para variados fins. [21] sugerem o casamento de imagens e reconhecimento de objetos, adotando, como hipótese, a permanência da organização estrutural de um objeto em movimento em intervalos de tempo suficientemente pequenos. Neste sentido, é desenvolvida uma função de distância entre duas árvores de confinamento [6], baseada na relação entre dupla de nós folhas e raiz. [22] propõem uma simplificação e verificação de isomorfismo de árvores de componentes para registro de imagens médicas topologicamente similares (duas fatias de imagens tomográficas do mesmo objeto, por exemplo). [5] propõe uma filtragem, com base na informação da árvore de componentes, utilizando um conceito denominado *assinatura de atributo*. Esta é definida pela combinação de atributos ditos planares (área, por exemplo) e não-planares (menor sequência de nós ligados partindo de um nó folha até a raiz, por exemplo) para discriminar, com sucesso, características em micrografias de madeira. [28] também elabora uma filtragem de atributo não-crescente (vide Eq. 2.12) da árvore por programação dinâmica [75] de propriedades geométricas de letras para detecção de textos das legendas em quaisquer imagens ou vídeos em níveis de cinza. Considera-se que as letras estão representadas nas folhas e em alguns de seus ancestrais consecutivos (supondo uso de Max-tree e também de Min-tree). Supressões de nós são feitas (não há preservação de nó, cujo pai deva ser removido) e, pela subtração (*top-hat*) entre a imagem original e a filtrada, espera-se recuperar as letras. Os atributos mais discriminantes, segundo os autores, foram a complexidade (razão entre perímetro e área) e a compacidade (área dividida pelo quadrado do perímetro), ambos não-crescentes. [12] relatam uma forma de segmentação de imagens através da obtenção dos máximos (folhas) da árvore de componentes após uma abertura por área e altura. [19], por sua vez, acrescentam os atributos volume e compacidade, para filtragem de imagens dermatológicas, e consequente segmentação de pintas na pele (melanócito nevos) a partir de uma análise estatística simples sobre os nós selecionados da árvore. [18] se restringem a sinais unidimensionais, propondo multilimiarização da árvore de componentes obtida eficientemente do histograma da imagem. Em relação à aplicação

de perseguição de objetos, [24] utilizam fluxo óptico para construção de um critério não-crescente do filtro morfológico de movimento [76] da Max-tree, e [26] detectam, com base no caminho de folhas à raiz desta árvore, regiões extremas maximalmente estáveis (MSER) [77]. [78] ainda propõem um método para detecção de quadros em vídeo com *flash* (alta luminosidade) baseado na busca de "montanhas" com alturas maiores a um certo limiar h pré-definido e a área da base, menor ou igual a outro valor S, correspondente à duração do *flash* (considerou-se $S = 5$ quadros como a máxima duração do flash), obtidos na filtragem da Max-tree de um sinal unidimensional proveniente da simplificação de ritmo visual da imagem [79]. A intenção dos autores é evitar que "clarões" em vídeo sejam encarados como transição entre cenas. Na presente tese, a Max-tree também é substrato para aplicações, descritas ao longo do texto, de filtragem e segmentação [17], simplificação de imagens [9] e reconhecimento de formas [29, 30].

3.3 Estado da arte

Nesta seção, algoritmos para construção eficiente da Max-tree são apresentados. Em relação a sinais unidimensionais, [18] determinam tal árvore em tempo linear, com consumo racional de memória, fazendo uso de uma pilha auxiliar para registrar o histórico pregresso de visitação dos pontos (pixels). Com isto, procedem-se relações de inclusão dos componentes em uma única varredura do sinal. A partir do valor b do predecessor do pixel visitado, pode-se tomar três decisões indicadas na Figura 3.4: **(i)** se for menor que o valor a do pixel visitado, cria-se novo nó; **(ii)** se igual a a, refere-se ao mesmo nó; **(iii)** se maior que a, sabe-se que o nó do predecessor já existe, restando definir se o pixel atual refere-se a um novo nó ou a um nó já criado e, para isto, o conteúdo da pilha é consultado. A pilha pode se configurar em quatro situações exibidas na Figura 3.4: **(iv)** se pilha vazia, um novo nó é criado para o pixel visitado, sendo este, pai do nó de seu predecessor; **(v)** se, na base da pilha (primeiro valor empilhado), tiver um valor c menor que o valor a do pixel atual visitado, o procedimento é o mesmo que o anterior; **(vi)** se o valor c for igual, então o pixel atual refere-se ao mesmo nó que este; **(vii)** se c for maior, então este é pai do predecessor do pixel atual, e o pixel atual (valor a) é candidato a ser pai do pixel da base (valor c). No algoritmo, um valor é empilhado sempre que maior que o anterior, e desempilhado assim que a relação pai-filho entre componentes passa a ser estabelecida.

No caso bidimensional, a partir do volume observado de citação pela comunidade acadêmica, dois algoritmos se destacam na implementação eficiente da árvore de componentes, um baseado em busca-união de [80] (*union-find-based*) [40, 81, 36] e outro, baseado em inundação hierárquica recursiva (*flooding-based*) [1, 32]. Estes são detalhados a seguir.

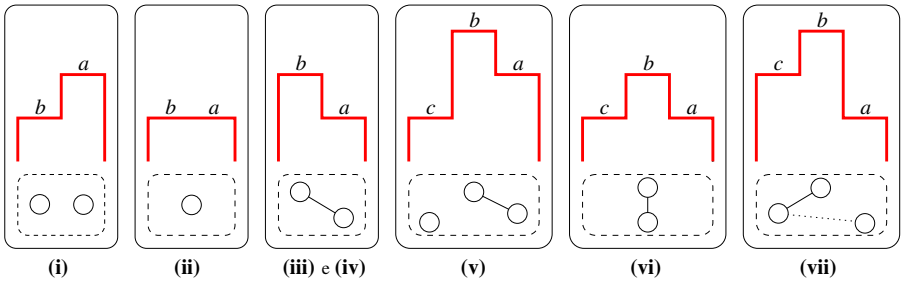


Fig. 3.4: Cenários possíveis para o valor do pixel atualmente visitado a, seu predecessor b e o primeiro pixel considerado c, com os nós da árvore definidos e ligações pai-filho estabelecidas.

3.3.1 Método baseado em *union-find*

Pode-se imaginar que a superfície topográfica, referente à imagem em níveis de cinza, esteja totalmente submersa. Na medida que esta água escoa, surgem ilhas, cujo topo correspondem aos máximos regionais. Supondo o escoamento constante da água, duas ou mais ilhas podem se unir (atribuição de um pai para dois ou mais filhos) sucessivamente até que haja apenas um bloco de "terras" (raiz da árvore). Este é o princípio de construção da árvore de componentes baseado em *union-find* [81, 36, 23]. Para implementação desta ideia, consideram-se os pixels na ordem decrescente de suas intensidades na imagem (intensidades maiores surgem como primeiras ilhas). Se, para o pixel atual, há um vizinho com intensidade igual, então este é unido à região do primeiro. Se há vizinho com intensidade maior, então é feita uma ligação pai-filho. Vizinhos com intensidades menores são desconsiderados até que sejam alcançados na sequência decrescente dos pixels. Para cada nó visitado da maior para menor intensidade, ou cria-se uma nova região, ou une-se a uma região existente, ou cria-se uma ligação entre uma região filho (mais alta intensidade) para uma região pai (mais baixa intensidade). Este processo se torna bastante eficiente se um algoritmo de *union-find* [80, 40] for aplicado, onde os conjuntos em questão são definidos pelas intensidades de cinza. E cada região ou nó é nomeado por uma intensidade. Se houver dois nós com mesmo valor, rótulos inteiros secundários são atribuídos a cada nó para evitar ambiguidades. A Figura 3.5 sintetiza algumas passagens do método para uma imagem com uma única linha.

3.3.2 Método baseado em *flooding*

Em vez do sentido topo até base da topografia, pode-se construir a mesma árvore no sentido inverso. Visualiza-se agora o relevo como uma grande caverna, modelado apenas como uma casca externa. Não mais se supõe a superfície totalmente submersa, mas a ocorrência de uma inundação (*flood*) especial (no sentido de agregação de pixels vizinhos), recursiva e hierárquica (chamada a própria função, se vizinho, em análise, apresentar nível de cinza superior) [1] que, para um entendi-

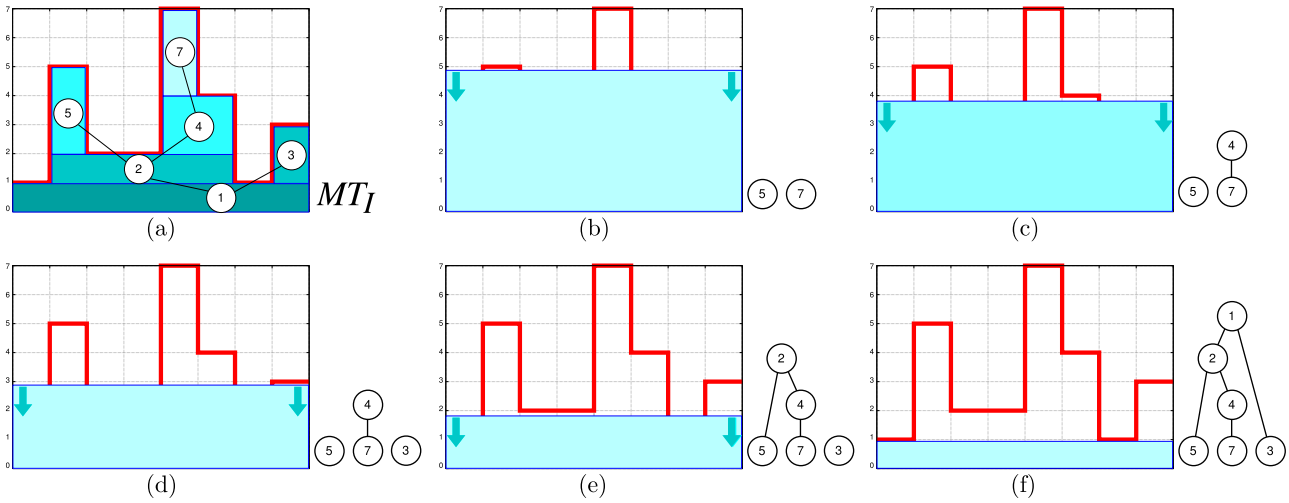


Fig. 3.5: Passagens do algoritmo baseado em *union-find* sobre $I(p) = \{1, 5, 2, 2, 7, 4, 1, 3\}$.

mento rápido inicial, pode ser interpretada da seguinte maneira: a partir da base da superfície (um dos pixels de menor intensidade), água passa a ser injetada até que surjam dois ou mais lagos, em dado momento, internos ao relevo. Bloqueia-se então a elevação de água de todos os lagos, exceto um (segundo a sequência de visitação dos pixels pré-definida). O nível de água aumenta apenas neste lago até o momento em que derivam-se dois ou mais lagos novamente. Repete-se tal processo (chamadas recursivas hierárquicas) até que não se possa mais injetar água (trata-se de um ponto de máximo regional). Inicia-se então a drenagem deste último lago interno considerado até uma altura (nível de cinza) tal que a área de sua lâmina de água se torne maior (finalização de chamadas recursivas). Neste ponto deve ser estabelecida uma ligação filho-pai. A região drenada é então bloqueada e marcada como concluída (pixels já visitados). Em um dos demais lagos bloqueados (mas não concluídos), repete-se o processo de injeção de água até o topo (máximo), e sua drenagem para o estabelecimento de relações pai-filhos entre os lagos. Após a remoção de água de duas ou mais "montanhas", um único lago interno pode surgir (atribuição de um pai a dois ou mais filhos) e assim, sucessivamente, até que haja uma lâmina de água contínua na base de toda a superfície (raiz da árvore). A implementação desta solução requer apenas a determinação de um pixel de intensidade mínima (não é necessária a ordenação como no método baseado em *union-find*) e se torna eficiente com a utilização de *hashing* estático para pronta localização das subárvores criadas em cada chamada recursiva [82]. A Figura 3.6 sintetiza algumas passagens do método para uma imagem com uma única linha.

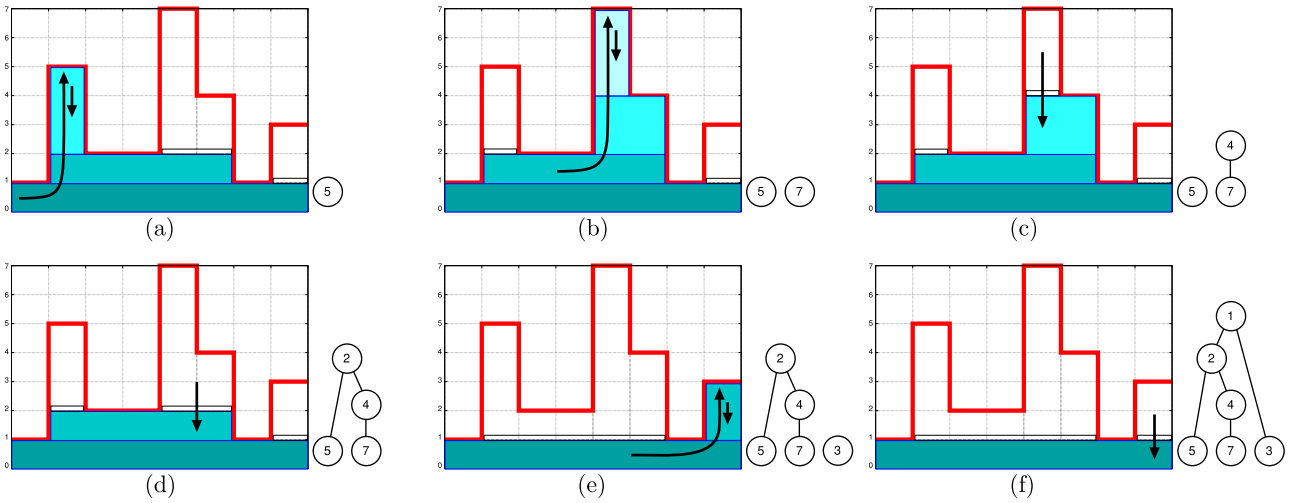


Fig. 3.6: Passagens do algoritmo baseado em *flooding* sobre $I(p) = \{1, 5, 2, 2, 7, 4, 1, 3\}$.

3.3.3 Discussão

As árvores de [1] e [36] diferem ligeiramente das estruturas de [5] e [35], por estes dois últimos considerarem todos os limiares possíveis na determinação dos componentes, enquanto os primeiros utilizam somente limiares iguais às intensidades de cinza que constituem zonas planas na imagem. Neste trabalho, optou-se por essa forma mais compacta que possibilita um processamento mais eficiente da árvore. De qualquer forma, a árvore, sendo criada a partir da decomposição por limiares, tem como altura topológica máxima o valor $L - 1$ (maior nível de cinza representável). No algoritmo de [1], cada pixel é inserido em uma fila hierárquica (múltiplas filas, uma para cada intensidade da imagem) uma única vez e seus k vizinhos (normalmente $k = 4$ ou $k = 8$), verificados para proceder ou não chamadas recursivas conforme a comparação entre os níveis de cinza, sendo necessárias, portanto, $k \times n$ comparações[4] para uma imagem com n pixels. A complexidade, neste caso, é de $O(L \times n)$, onde L é o número de níveis de cinza da imagem. O método de [36], baseado na estrutura *union-find* exposta acima, modela a imagem como um grafo, e determina a árvore em tempo quase-linear $O(n \times \alpha(n))$, onde n refere-se ao número de vértices mais arestas, e $\alpha(n)$ à função inversa de Ackermann com contradomínio de valor máximo bastante reduzido (na prática, $\alpha(n) < 4$) [80]. [23], tomando como base esta solução, se concentra em otimizações para obtenção de ganho de desempenho na construção da árvore para imagens com alta quantização (como, por exemplo, fotos astronômicas de 16 bits), e uso racional de memória na representação da estrutura para este tipo de imagem (aplicações de astronomia, por exemplo, exigem considerável quantidade de memória). O algoritmo proposto por [23] requer uso de memória significativamente menor comparado à proposta de

[4]Para um cálculo mais preciso, deve-se desconsiderar vizinhos fora do domínio da imagem.

[36] e, além disto, se mostra como melhor solução, em relação ao tempo de execução, para imagens de 16 bits [23]. A Max-tree baseada em *union-find*, seja pelo procedimento original ou otimizado, é apropriada para imagens com intensidades no domínio real. Ao utilizar *flooding*, a extensão para implementação de recurso semelhante despenderia custo proibitivo de memória, na medida que há menor probabilidade de repetição de números reais e, consequentemente, um número elevado de entradas na fila hierárquica seria possivelmente necessário. Uma solução para contornar tal problema pode vir da transformação da escala dinâmica da imagem e seu arredondamento para inteiro com maior representação binária. Ainda assim, a quantidade de níveis pode ser elevada. Todavia, este trabalho se restringe a imagens inteiras (vide Def. 2.1). A Max-tree baseada em *flooding*, para imagens inteiras com L relativamente pequeno (normalmente 256), é construída em cerca de 2 vezes menos tempo que sua versão utilizando *union-find* [36].

Como visto na Seção 3.2, as aplicações de Max-tree são bastante variadas. Um ou mais operadores podem ser implementados a partir de uma mesma estrutura de dados criada. Porém, muitas vezes, há interesse apenas no resultado de um processamento específico. Neste sentido, uma abordagem é a de utilizar o algoritmo de construção da árvore, mas não alocar, de fato, sua estrutura de dados, devido ao foco estar na geração imediata de resultados de uma ação específica. Neste caso, não é possível, por exemplo, uma eventual reutilização da árvore filtrada por outros operadores. [15] seguem esta linha e propõem um método de abertura e fechamento por atributo, a partir da modificação do algoritmo de [1], porém sem construção efetiva da árvore.

O presente trabalho se concentra na persistência da árvore para processamentos simultâneos ou sucessivos, evidenciando os ganhos em desempenho que esta adoção acarreta.

3.4 Algoritmo proposto

A proposta de algoritmo para a construção da árvore de componentes deste trabalho é baseada no pseudocódigo apresentado por [1], devido a sua implementação simples e por ter se demonstrado mais eficiente na prática (uma comparação de desempenho será mostrada a seguir) para imagens inteiras. Para a localização dos nós em memória, de modo que as ligações "pai-filho" possam ser efetuadas, é utilizado um *hashing*, de forma semelhante ao trabalho de [82]. A principal contribuição da proposta consiste na inserção de novos atributos calculados incrementalmente no processo de construção da Max-tree, e o uso de tabelas de nós para que se possa reduzir significativamente o número de alocações sucessivas e, consequentemente, o tempo computacional desta atividade. O tamanho destas tabelas é constante, ajustado para evitar que a memória extra não utilizada seja mínima. Apenas imagens com níveis de cinza pertencentes aos números naturais (conforme Def. 2.1), podem ser utilizadas, tornando a solução relativamente genérica (dada a possibilidade de discretização e nor-

malização mesmo de um domínio real), rápida (pelo uso de estruturas estáticas) e previsível (no que diz respeito aos recursos de *hardware* como uso máximo de memória, por exemplo). A restrição de domínio das intensidades, porém, não compromete o processamento de imagens adquiridas por dispositivos usuais (câmeras ou *scanners* pessoais ou industriais, por exemplo), sendo a estrutura, de fato, apropriada a este tipo de imagem e suficiente a uma gama de aplicações, algumas exemplificadas neste trabalho. Segue o detalhamento destas colocações.

3.4.1 Novos atributos

A presente proposta trata-se de uma extensão do algoritmo baseado em *flooding* de [1]. As linhas iniciadas com "►", no Algoritmo 3.1 apresentado logo a seguir, referem-se aos passos auxiliares para a determinação incremental de novos atributos para cada nó da Max-tree, culminando com novas informações para descrição mais detalhada dos componentes de nível da imagem. Em termos de implementação, em linguagem de programação (no caso, C++), alguns refinamentos são sugeridos como uso de filas hierárquicas estáticas para a inundação eficiente, além de um *hashing* baseado no histograma da imagem para auxiliar na localização imediata, em memória, dos nós da árvore. Esta localização é útil para a junção de nós (estabelecimento de ligação "pai-filho") no processo de construção da Max-tree ou, uma vez finalizada esta etapa, para a simples consulta dos atributos associados a um componente de nível. Atributos bem estabelecidos na literatura, como **área**, **altura** e **volume**, continuam sendo determinados, pois são importantes no projeto de filtros. Atributos adicionais, também calculados de forma incremental no processo de construção da Max-tree – ou seja, mantendo a complexidade do algoritmo original de [1] –, são descritos mais à frente. Antes, uma visão geral das estruturas é apresentada.

Modelagem das estruturas. Para a construção e manipulação da Max-tree, sugerem-se as estruturas de dados da Figura 3.7 (representando a imagem da Fig. 3.1) com os nós constituintes – cada um com referências (ligações) ao filho (mais velho), ao irmão (imediatamente mais novo), ao pai e ao próximo da sequência de inserção (uma lista encadeada é definida, de forma lógica, para visitação simples de todos os nós) –, com a árvore em si, e com uma tabela *hashing* para cada nível de cinza com objetivo de localização eficiente de um nó em particular (em memória). Na figura, os nós ainda carregam dois atributos – nível de cinza e rótulo do componente de nível associado. Porém, cada nó armazena uma série de outros atributos geométricos/estatísticos (domínio da imagem) ou topológicos (domínio da árvore). A composição dos nós e demais elementos serão detalhados no restante da seção.

O Algoritmo 3.1 implementa parte (o Alg. 3.2 o complementa) do cálculo dos atributos propostos. O método, baseado em inundação hierárquica de [1], em função da sequência de pixels p

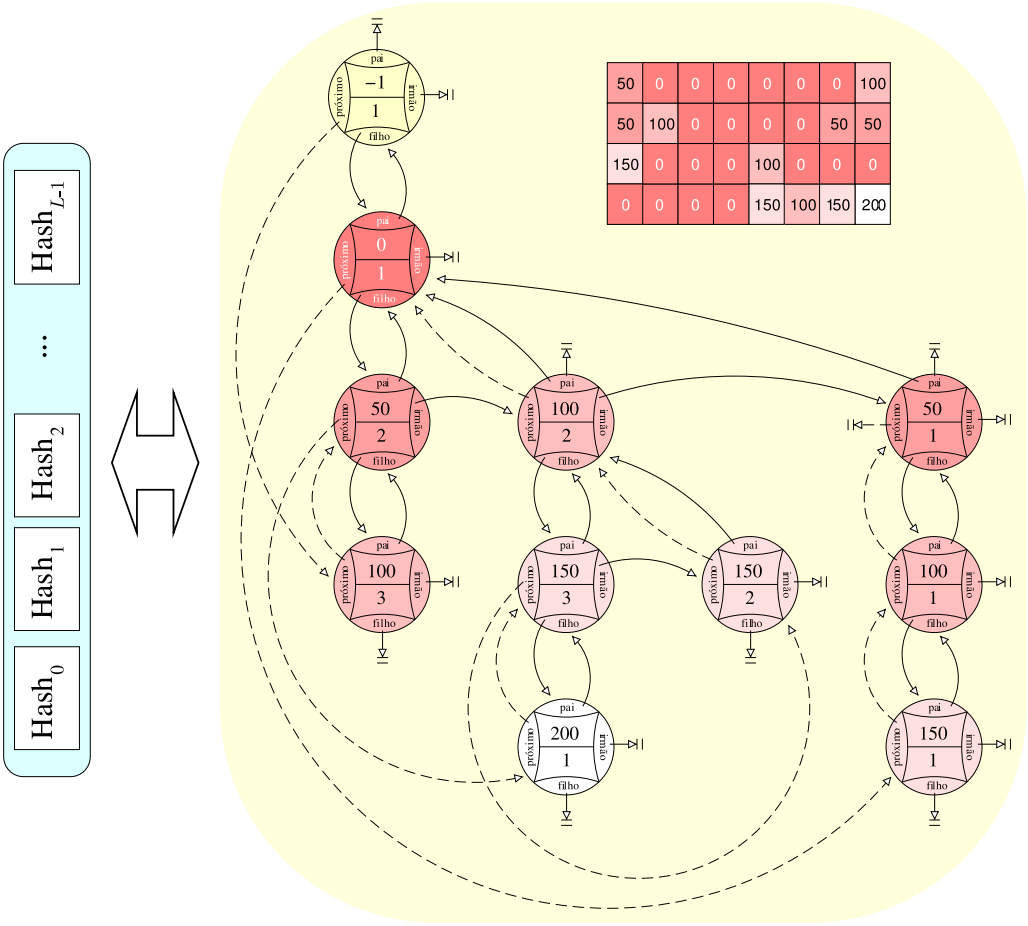


Fig. 3.7: Exemplo sintético da organização adotada de estrutura do nó (com nível e rótulo), conjunto de ligações para descrição da árvore, e *hashing* de localização de nós em cada nível de cinza.

visitados (linha 13) para o escopo recursivo de cada nível de cinza (linha 25), é utilizado, acrescentando, a cada componente de nível, a determinação de coordenadas de seu centroide (linhas 10 e 14), as coordenadas do canto superior esquerdo $x_{\ulcorner}$ e inferior direito $x_{\lrcorner}$ de sua caixa envolvente, além de estatísticas dos níveis de cinza na imagem em seu domínio, como média n_μ, desvio padrão n_σ, intensidades mínima n_{min} e máxima n_{max}. LINK (linhas 30 e 32) estabelece a conexão entre os nós pai e filho representados por um conjunto básico (nível de cinza, rótulo e coordenada da semente) e demais valores ($x_{\ulcorner}$, $x_{\lrcorner}$, x_c, $area$ e conjunto estatístico $estat$) que devem ser atualizados, considerando que os pixels do filho estão incluídos no componente de seu pai. O Algoritmo 3.2 detalha esta conexão, complementa a atualização destes atributos, e adiciona informações topológicas para cada nó, como a altura de sua subárvore ($htop$), e a quantidade de filhos ($grau$) e de descendentes ($desc$).

Atributos no domínio da imagem. Alguns atributos estão relacionados ao conjunto de pixels no domínio E da imagem. O Algoritmo 3.1 determina (marcações "▶" em vermelho), para cada com-

ponente de nível, as seguintes coordenadas geométricas:

- **Coordenada da semente de reconstrução.** Mantém-se agora um único pixel semente pertencente a uma zona plana de componente de nível $\mathcal{C}_n^r(I)$ para sua ágil reconstrução, caso haja esta necessidade após a finalização da construção da Max-tree MT_I, com base apenas na imagem I, no nível n e rótulo r (assim como visto na Fig. 3.2). Para tanto, um vetor auxiliar $coord$ registra a coordenada do último pixel visitado (sua criação e atualização podem ser conferidas nas linhas 3, 23 e 34 do Alg. 3.1).

- **Coordenada do centroide.** Persiste-se o pixel referente ao centroide x_c de $\mathcal{C}_n^r(I)$ (linhas 10 e 14 do Alg. 3.1). Ao final da construção da Max-tree, tem-se, de fato, as soma das linhas e soma das colunas dos pixels do componente e, portanto, faz-se necessária a divisão inteira por sua área (número de pixels) para correta utilização (vide Eq. 2.15 e 2.23 do Cap. 2), conforme mostrada na complementação (linha 36).

- **Coordenadas do canto superior esquerdo e inferior direito da caixa envolvente.** Outras duas informações determinadas no processo de inundação são as coordenadas $x_{\ulcorner}$ e $x_{\lrcorner}$, necessárias à definição de um retângulo com dimensões mínimas para envolver $\mathcal{C}_n^r(I)$ completamente (linhas 10 e $15-16$ do Alg. 3.1). Os símbolos $\succ$ ou $\prec$ comparam (maior ou menor) ambas coordenadas (linha **e** coluna) na determinação dos cantos da caixa envolvente.

Bem como os seguintes cálculos estatísticos (marcações "▶" em verde, com inicializações na linha 11 do Alg. 3.1) sobre os níveis de cinza da imagem original, restrito ao domíno de pixels de cada componente de nível:

- **Média.** Obtém-se a média n_μ dos níveis de cinza do componente procedendo, a cada pixel visitado (linha 13), a soma acumulada da intensidade na imagem (linha 17 do Alg. 3.1) e, no final, a divisão pela área (linha 38 do Alg. 3.1).

- **Desvio padrão.** A dispersão n_σ dos pixels do componente, dada pela normalização da soma do quadrado das diferenças entre níveis de cinza e média (Eq. 2.24), também pode ser determinada incrementalmente, a partir de pequena manipulação no cálculo da variância: $n_\sigma^2 = \frac{\sum_{i=1}^{area}(n_i-n_\mu)^2}{area-1} = \frac{\sum_{i=1}^{area}(n_i^2)-area\cdot n_\mu^2}{area-1} = \frac{1}{area-1}\left(\sum_{i=1}^{area}(n_i^2) - \frac{\left(\sum_{i=1}^{area}(n_i)\right)^2}{area}\right)$. Os dois somatórios da última expressão (igualdade mais à direita) são determinados na linha 17 do Algoritmo 3.1. E, no final (linha 38), o desvio padrão, conforme a equação, é obtido de fato.

- **Máximo e mínimo.** Também são registradas as intensidades mínima n_{min} (igual ao nível do nó) e máxima n_{max} (linha 18 do Alg. 3.1) da imagem no domínio de cada componente de nível.

Algoritmo 3.1: Extensão da construção da Max-tree de [1] para determinação incremental de novos atributos geométricos, estatísticos e topológicos.

ENTRADA: $I, \mathcal{V}_E$
SAÍDA: MT_I

INICIALIZAÇÃO:
1 $level[k] \leftarrow$ **falso** , $\forall k \in [0, L-1]$ *//níveis de cinza atuais*
2 $label[k] \leftarrow 0, \forall k \in [0, L-1]$ *//rótulos atuais*
3 ▶ $coord[k] \leftarrow \{0,0\}, \forall k \in [0, L-1]$ *//sementes de reconstrução atuais*
4 $queue[k] \leftarrow \emptyset, \forall k \in [0, L-1]$ *//filas hierárquicas*
5 $queue[\min(I)]$.insere(x_m) tal que $I(x_m) = \min(I)$
6 $status[x] \leftarrow 0, \forall x \in E \subset \mathbb{N}^2$ *//rótulos persistentes*
7 $MT_I \leftarrow \emptyset$
8 FLOOD$(\min(I))$
9 COMPLEMENTAÇÃO()

FLOOD(n)
10 ▶ $area \leftarrow 0$; $x_c \leftarrow \{0,0\}$ $x_\ulcorner \leftarrow \{\infty,\infty\}$; $x_\lrcorner \leftarrow \{0,0\}$ *//área, centroide e coordenadas da caixa envolvente*
11 ▶ $estat \leftarrow \{n_\mu, n_\sigma, n_{max}\} \leftarrow \{0,\ 0,\ 0\}$ *//média, dispersão e máximo das intensidades*
12 **enquanto** $queue[n] \neq \emptyset$
13 $p \leftarrow queue[n]$.remove()
14 ▶ $area \leftarrow area + 1$; $x_c \leftarrow x_c + p$
15 ▶ **se** $p \lessdot x_\ulcorner$ $\Rightarrow$ $x_\ulcorner \leftarrow p$
16 ▶ **se** $p \gtrdot x_\lrcorner$ $\Rightarrow$ $x_\lrcorner \leftarrow p$
17 ▶ $n_\mu \leftarrow n_\mu + I[p]$; $n_\sigma \leftarrow n_\sigma + I[p]^2$
18 ▶ **se** $I[p] > n_{max}$ $\Rightarrow$ $n_{max} \leftarrow I[p]$
19 $status[p] \leftarrow label[n] + 1$
20 ▶ **para cada** $q \in \mathcal{V}_E(p)$
21 **se** $status[q] = 0$ *//não analisado*
22 $m \leftarrow I[q]$; $queue[m]$.insere(q); $level[m] \leftarrow$ **verdadeiro** ; $status[q] \leftarrow -1$ *//na fila*
23 ▶ $coord[m] \leftarrow q$ *//nova semente de reconstrução*
24 **enquanto** $m > n$
25 $m \leftarrow$ FLOOD(m)
26 $m \leftarrow n - 1$
27 **enquanto** $m \geq 0$ **e** (**não** $level[m]$)
28 $m \leftarrow m - 1$
29 **se** $m \geq 0$
30 ▶ MT_I.LINK$(\{m, label[m]+1, coord[m]\}, \{n, label[n]+1, coord[n]\}, x_\ulcorner, x_\lrcorner, x_c, area, estat)$
31 **senão**
32 ▶ MT_I.LINK$(\{-1, 1, -1\}, \{\min(I), 1, coord[\min(I)]\}, x_\ulcorner, x_\lrcorner, x_c, area, estat)$ *//nó-cabeça*
33 $level[n] \leftarrow$ **falso** ; $label[n] \leftarrow label[n] + 1$;
34 ▶ $coord[n] \leftarrow p$
35 **retorne** m

COMPLEMENTAÇÃO()
36 **para cada** $\mathcal{N} \in MT_I$ (percurso em largura)
37 ▶ $\mathcal{N}_{x_c} \leftarrow \mathcal{N}_{x_c}$ div $\mathcal{N}_{area}$
38 ▶ $\mathcal{N}_{n_\sigma} \leftarrow \left[\frac{1}{\mathcal{N}_{area}-1}\left(\mathcal{N}_{n_\sigma} - \frac{\mathcal{N}^2_{n_\mu}}{\mathcal{N}_{area}}\right)\right]^{\frac{1}{2}}$ se $\mathcal{N}_{area} > 1$; $\mathcal{N}_{n_\sigma} = 0$ c.c.; $\mathcal{N}_{n_\mu} \leftarrow \frac{\mathcal{N}_{n_\mu}}{\mathcal{N}_{area}}$
39 **para cada** $\mathcal{N}^F$ filho de $\mathcal{N}$
40 $\mathcal{N}^F_{ntop} \leftarrow \mathcal{N}_{ntop} + 1$

Atributos no domínio da árvore. Alguns atributos referem-se à topologia de estrutura hierárquica (vide Sec. 2.4). Os mesmos são atualizados, portanto, no momento em que dois nós estabelecem ligação do tipo "pai-filho" (chamada à função LINK nas linhas 30 e 32):

- **Nível topológico.** Mantém-se a informação do comprimento do caminho (número de arcos ou ligações) de $\mathcal{N}_{\mathcal{C}_n^r}$ até a raiz $\mathcal{N}^R$. A raiz $\mathcal{N}^R$ tem, portanto, nível topológico 0, os filhos de $\mathcal{N}^R$ têm nível topológico 1, os netos de $\mathcal{N}^R$, nível 2, e assim por diante.

- **Altura topológica.** Considerando apenas a subárvore enraizada em $\mathcal{N}_{\mathcal{C}_n^r}$, trata-se do comprimento máximo entre todos os caminhos orientados possíveis partindo de $\mathcal{N}_{\mathcal{C}_n^r}$, ou seja, o comprimento do caminho de $\mathcal{N}_{\mathcal{C}_n^r}$ ao seu descendente de maior nível topológico (mais profundo). Outra designação comum para esta definição é *altura da subárvore*.

- **Grau.** Aproveita-se a inundação para se atualizar também o atributo que registra o número de filhos de $\mathcal{N}_{\mathcal{C}_n^r}$, assim que uma nova ligação "pai-filho" ocorre neste nó.

- **Número de descendentes.** O último atributo topológico, determinado em tempo de construção da árvore, é o número de descendentes de $\mathcal{N}_{\mathcal{C}_n^r}$. Sempre que uma nova ligação "pai-filho" é estabelecida, todos os ancestrais acrescentam 1 a este número.

O Algoritmo 3.2 detalha esta ligação de nós "pai-filho" (linhas 30 ou 32 do Alg. 3.1), conforme a inundação hierárquica, com a determinação da altura topológica, grau e número de descendentes de forma incremental – para atualização do nível topológico, um percurso em profundidade, em tempo linear, é feito posteriormente (linhas $39-40$ do Alg. 3.1). Primeiramente, uma busca pelos nós pai e filho é feita através de uma tabela *hashing* auxiliar (linhas 1-2). Se um destes dois nós não é encontrado, então deve ser criado por NEW_NODE (linha 4 ou 6 ou 19). Há, portanto, quatro possibilidades: **Inexistência de ambos nós, pai e filho** – A busca na tabela *hashing* (linhas 1-2) indica que não há referência aos dois nós. Ambos devem então ser alocados (linhas 4 e 6), a lista encadeada de nós atualizada (linha 7), assim como os atributos volume, altura e número de descendentes do pai (linhas $8-9$) –; **Existência apenas do nó filho** – O pai deve ser alocado (linha 4) e os atributos de caixa envolvente, centroide, área, altura, volume, número de descendentes ($desc$) e altura topológica ($htop$), atualizados (linhas $11-16$) –; **Existência apenas do nó pai** – O filho deve ser alocado (linha 19) e os atributos, atualizados (linhas $20-25$). Observar que, já havendo o nó pai, eventualmente contido em uma árvore, tem-se que os números de descendentes de todos os seus ancestrais devem acrescer em uma unidade (linhas $23-25$) pela entrada deste novo nó filho (descendente) –; **Existência de ambos nós, pai e filho** – Por fim, os dois nós podem já ter sido criados e eventualmente serem parte de árvores temporárias (no processo de construção, pode-se constituir uma floresta). Estes têm todos os seus atributos atualizados (linhas $27-33$). O processo finaliza com a atualização também

dos atributos estatísticos (linhas $34-35$), e ligações "irmão-irmão" e "pai-filho" (linha 36), de forma a ir estruturando a Max-tree.

Algoritmo 3.2: Algoritmo para inserção do relacionamento "pai-filho" e cálculo de novos atributos nos domínios da imagem e da árvore.

ENTRADA: $MT_I, \{n_{pai}, r_{pai}, x_{pai}\}, \{n_{filho}, r_{filho}, x_{filho}\}, x_{\ulcorner}, x_{\lrcorner}, x_c, area, \{n_\mu, n_\sigma, n_{max}\}$
SAÍDA: MT_I

LINK()
1 $\mathcal{N}^P \leftarrow MT_I$.HASH_TABLE($\{n_{pai}, r_{pai}\}$)
2 $\mathcal{N}^F \leftarrow MT_I$.HASH_TABLE($\{n_{filho}, r_{filho}\}$)
3 **se** $\nexists \mathcal{N}^P$
4 $\quad \mathcal{N}^P \leftarrow MT_I$.NEW_NODE($n_{pai}, r_{pai}, x_{pai}, x_{\ulcorner}, x_{\lrcorner}, x_c, area, n_\mu, n_\sigma, n_{max}$)
5 $\quad$ **se** $\nexists \mathcal{N}^F$
6 $\quad\quad \mathcal{N}^F \leftarrow MT_I$.NEW_NODE($n_{filho}, r_{filho}, x_{filho}, x_{\ulcorner}, x_{\lrcorner}, x_c, area, n_\mu, n_\sigma, n_{max}$)
7 $\quad\quad$ próximo de $\mathcal{N}^F \leftarrow \mathcal{N}^P$
8 $\quad\quad \mathcal{N}^P_{volume} \leftarrow \mathcal{N}^P_{volume} + area \cdot (n_{filho} - n_{pai})$; $\mathcal{N}^P_{altura} \leftarrow n_{filho} - n_{pai}$
9 $\quad\quad \mathcal{N}^P_{desc} \leftarrow \mathcal{N}^P_{desc} + 1$; $\mathcal{N}^P_{htop} \leftarrow \mathcal{N}^P_{htop} + 1$
10 $\quad$ **senão**
11 $\quad\quad$ **se** $x_{\ulcorner} \precsim \mathcal{N}^F_{x_{\ulcorner}} \Rightarrow \mathcal{N}^F_{x_{\ulcorner}} \leftarrow x_{\ulcorner}$; **se** $x_{\lrcorner} \succsim \mathcal{N}^F_{x_{\lrcorner}} \Rightarrow \mathcal{N}^F_{x_{\lrcorner}} \leftarrow x_{\lrcorner}$
12 $\quad\quad$ **se** $\mathcal{N}^F_{x_{\ulcorner}} \precsim \mathcal{N}^P_{x_{\ulcorner}} \Rightarrow \mathcal{N}^P_{x_{\ulcorner}} \leftarrow \mathcal{N}^F_{x_{\ulcorner}}$; **se** $\mathcal{N}^F_{x_{\lrcorner}} \succsim \mathcal{N}^P_{x_{\lrcorner}} \Rightarrow \mathcal{N}^P_{x_{\lrcorner}} \leftarrow \mathcal{N}^F_{x_{\lrcorner}}$
13 $\quad\quad \mathcal{N}^F_{cent} \leftarrow \mathcal{N}^F_{cent} + x_c$; $\mathcal{N}^P_{cent} \leftarrow \mathcal{N}^F_{cent} + 1$; $\mathcal{N}^F_{area} \leftarrow \mathcal{N}^F_{area} + area$
14 $\quad\quad \mathcal{N}^P_{area} \leftarrow \mathcal{N}^F_{area}$; $\mathcal{N}^P_{altura} \leftarrow \mathcal{N}^F_{altura} + (n_{filho} - n_{pai})$
15 $\quad\quad \mathcal{N}^P_{volume} \leftarrow \mathcal{N}^P_{volume} + \mathcal{N}^F_{volume} + \mathcal{N}^F_{area} \cdot (n_{filho} - n_{pai})$
16 $\quad\quad \mathcal{N}^P_{desc} \leftarrow \mathcal{N}^P_{desc} + \mathcal{N}^F_{desc} + 1$; $\mathcal{N}^P_{htop} \leftarrow \mathcal{N}^P_{htop} + \mathcal{N}^F_{htop} + 1$;
17 **senão**
18 $\quad$ **se** $\nexists \mathcal{N}^F$
19 $\quad\quad \mathcal{N}^F \leftarrow MT_I$.NEW_NODE($n_{filho}, r_{filho}, x_{filho}, x_{\ulcorner}, x_{\lrcorner}, x_c, area, n_\mu, n_\sigma, n_{max}$)
20 $\quad\quad$ **se** $x_{\ulcorner} \precsim \mathcal{N}^P_{x_{\ulcorner}} \Rightarrow \mathcal{N}^P_{x_{\ulcorner}} \leftarrow x_{\ulcorner}$; **se** $x_{\lrcorner} \succsim \mathcal{N}^P_{x_{\lrcorner}} \Rightarrow \mathcal{N}^P_{x_{\lrcorner}} \leftarrow x_{\lrcorner}$
21 $\quad\quad \mathcal{N}^P_{cent} \leftarrow \mathcal{N}^P_{cent} + x_c$; $\mathcal{N}^P_{area} \leftarrow \mathcal{N}^P_{area} + area$; $\mathcal{N}^P_{volume} \leftarrow \mathcal{N}^P_{volume} + area \cdot (n_{filho} - n_{pai})$
22 $\quad\quad$ **se** $\mathcal{N}^P_{altura} < (n_{filho} - n_{pai}) \Rightarrow \mathcal{N}^P_{altura} = n_{filho} - n_{pai}$
23 $\quad\quad \mathcal{N}^P_{desc} \leftarrow \mathcal{N}^P_{desc} + 1$; $\mathcal{N}^A \leftarrow$ pai de $\mathcal{N}^P$
24 $\quad\quad$ **enquanto** $\mathcal{N}^A$
25 $\quad\quad\quad \mathcal{N}^A_{desc} \leftarrow \mathcal{N}^A_{desc} + 1$; $\mathcal{N}^A \leftarrow$ pai de $\mathcal{N}^A$
26 $\quad$ **senão**
27 $\quad\quad$ **se** $x_{\ulcorner} \precsim \mathcal{N}^F_{x_{\ulcorner}} \Rightarrow \mathcal{N}^F_{x_{\ulcorner}} \leftarrow x_{\ulcorner}$; **se** $x_{\lrcorner} \succsim \mathcal{N}^F_{x_{\lrcorner}} \Rightarrow \mathcal{N}^F_{x_{\lrcorner}} \leftarrow x_{\lrcorner}$
28 $\quad\quad$ **se** $\mathcal{N}^F_{x_{\ulcorner}} \precsim \mathcal{N}^P_{x_{\ulcorner}} \Rightarrow \mathcal{N}^P_{x_{\ulcorner}} \leftarrow \mathcal{N}^F_{x_{\ulcorner}}$; **se** $\mathcal{N}^F_{x_{\lrcorner}} \succsim \mathcal{N}^P_{x_{\lrcorner}} \Rightarrow \mathcal{N}^P_{x_{\lrcorner}} \leftarrow \mathcal{N}^F_{x_{\lrcorner}}$
29 $\quad\quad \mathcal{N}^F_{cent} \leftarrow \mathcal{N}^F_{cent} + x_c$; $\mathcal{N}^P_{cent} \leftarrow \mathcal{N}^P_{cent} + \mathcal{N}^F_{cent}$; $\mathcal{N}^F_{area} \leftarrow \mathcal{N}^F_{area} + area$
30 $\quad\quad \mathcal{N}^P_{area} \leftarrow \mathcal{N}^P_{area} + \mathcal{N}^F_{area}$; $\mathcal{N}^P_{volume} \leftarrow \mathcal{N}^P_{volume} + \mathcal{N}^F_{volume} + \mathcal{N}^F_{area} \cdot (n_{filho} - n_{pai})$
31 $\quad\quad \mathcal{N}^P_{desc} \leftarrow \mathcal{N}^P_{desc} + \mathcal{N}^F_{desc} + 1$
32 $\quad\quad$ **se** $\mathcal{N}^P_{htop} \leq \mathcal{N}^F_{htop} \Rightarrow \mathcal{N}^P_{htop} = \mathcal{N}^F_{htop} + 1$
33 $\quad\quad$ **se** $\mathcal{N}^P_{altura} < (n_{filho} - n_{pai}) \Rightarrow \mathcal{N}^P_{altura} = n_{filho} - n_{pai}$
34 $\mathcal{N}^F_{n_\mu} \leftarrow \mathcal{N}^F_{n_\mu} + n_\mu$; $\mathcal{N}^P_{n_\mu} \leftarrow \mathcal{N}^P_{n_\mu} + \mathcal{N}^F_{n_\mu}$ $\mathcal{N}^F_{n_\sigma} \leftarrow \mathcal{N}^F_{n_\sigma} + n_\sigma$; $\mathcal{N}^P_{n_\sigma} \leftarrow \mathcal{N}^P_{n_\sigma} + \mathcal{N}^F_{n_\sigma}$
35 $n_{max} > \mathcal{N}^F_{n_{max}} \Rightarrow \mathcal{N}^F_{n_{max}} \leftarrow n_{max}$; **se** $\mathcal{N}^F_{n_{max}} > \mathcal{N}^P_{n_{max}} \Rightarrow \mathcal{N}^P_{n_{max}} \leftarrow \mathcal{N}^F_{n_{max}}$
36 irmão de $\mathcal{N}^F \leftarrow$ filho de $\mathcal{N}^P$; pai de $\mathcal{N}^F \leftarrow \mathcal{N}^P$; filho de $\mathcal{N}^P \leftarrow \mathcal{N}^F$; $\mathcal{N}^P_{grau} \leftarrow \mathcal{N}^P_{grau} + 1$;

3.4.2 Estruturas de dados

Esta subseção esclarece a forma como se dá a implementação de filas hierárquicas (linhas $4-5$, $12-13$ e 22 do Alg. 3.1) e *hashing* (linhas $1-2$ do Alg. 3.2) para melhorar substancialmente o desempenho da construção da Max-tree.

Filas hierárquicas. O Algoritmo 3.1 faz uso de uma fila hierárquica auxiliar, ou seja, uma fila é definida para cada nível de cinza que ocorre na imagem, de modo a possibilitar a inundação recursiva ilustrada na Subseção 3.3.2. Caso a imagem tratada tenha quantidade considerável de componentes, também é acentuada a quantidade de inserções (linha 22) e remoções (linha 13) de pixels (coordenadas). O uso de estruturas dinâmicas não é a melhor solução, considerando o custo computacional de alocação de memória a cada inserção. Deste modo, optou-se por estabelecer filas estáticas circulares (*circular buffer*), nas quais, a reserva de memória é feita uma única vez para o vetor de pixels de cada fila. Porém, o tamanho deste vetor deve ser definido previamente e a solução foi a de utilizar o histograma para o nível de cinza em questão (vide Eq. 2.2), ou seja, a fila n ($queue_n$) deve ter espaço para entrada de, no máximo, $h_I(n)$ coordenadas de pixels, de acordo com a Figura 3.8. Sendo a posição FIM (do último pixel a ter entrado na fila) determinada, de forma lógica, para a circularidade apresentada, pela operação de resto da divisão inteira: $FIM \,\%\, h_I(n)$. A soma total dos tamanhos dos vetores é igual ao número total de pixels da imagem, ou $h_I(0) + h_I(1) + \cdots + h_I(L-1) = |E|$, com uso total de memória referente a dois inteiros (linha e coluna do pixel) vezes o número de pixels da imagem (cardinalidade do domínio ou $|E|$). É claro que, havendo mais de um componente com mesmo n, a fila neste nível não será preenchida totalmente em nenhum momento. O vetor de $queue_n$, tendo $h_I(n)$ posições, é suficientemente grande para comportar os pixels de todos os componentes com nível n, porém remoções ocorrem no processo de inundação, supondo a existência de regiões separando tais componentes. O desperdício de memória, no entanto, não é expressivo (a Seção 3.5 exibe comparações de uso de memória), sendo compensado por esta organização no que diz respeito ao consumo de recurso previsível (a fila hierárquica ocupa aproximadamente quatro vezes a memória de uma imagem de 8 bits de entrada) e velocidade maior de execução.

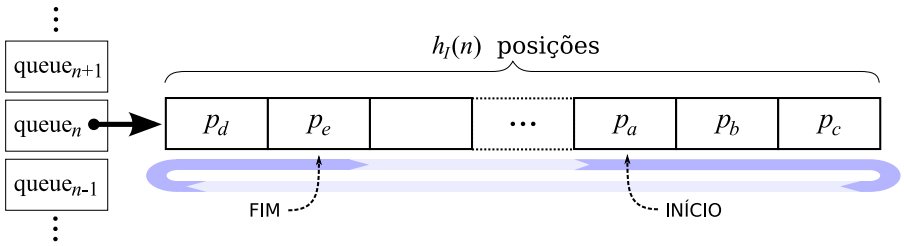


Fig. 3.8: Filas estáticas circulares de pixels para cada nível de cinza (hierárquicas).

***Hashing*.** O *hashing* utilizado no mapeamento de um limiar n e um rótulo r para localização direta (e imediata) de um nó $\mathcal{N}_{\mathcal{C}_n^r}$, em memória, de forma a obter todas as suas informações pertinentes, como relacionamentos "pai-filho", e atributos do componente de nível $\mathcal{C}_n^r$, necessário ao algoritmo de construção da árvore (linhas $1-2$ do Alg. 3.2), é também desenvolvido a partir do histograma h_I da imagem I. Como já visto, apenas as intensidades presentes em zonas planas da imagem geram nós da Max-tree. No entanto, para cada intensidade n, a limiarização $X_n(I)$ pode conter mais de um componente de nível rotulado devidamente por r. Em um caso extremo (hipotético), a limiarização $X_n(I)$ deve produzir $h_I(n)$ nós diferentes, supondo cada componente conexo com um único pixel. Neste sentido, a tabela *hashing* do nível n ($Hash_n$), deve ter $h_I(n)$ entradas para suportar este número máximo de nós, conforme ilustrado na Figura 3.9. Novamente, trata-se de um tamanho fixo pois a estrutura é estática e, portanto, as posições finais de cada tabela (vetor) podem eventualmente não ser utilizadas (ter referências nulas). No entanto, tal modelo permite um acesso extremamente rápido às informações de um nó $\mathcal{N}_{\mathcal{C}_n^r}$ pela obtenção de seu endereço em $Hash_n[r-1]$.

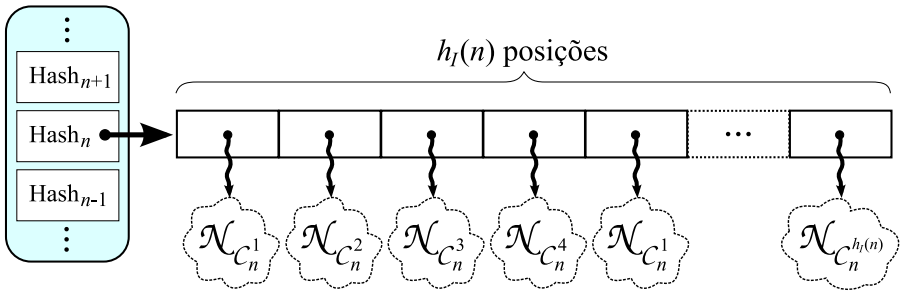


Fig. 3.9: *Hashing*, para cada nível de cinza n, de referências (endereços de memória) de nós $\mathcal{N}_{\mathcal{C}_n^r}$.

***Banco de nós*.** A cada novo nó (linhas 4, 6 e 19 do Alg. 3.2), deve haver memória para todos os seus atributos e suas referências (aos nós pai, filho mais velho, irmão mais novo e próximo). No entanto, também por questões de desempenho, a alocação não é feita individualmente, mas em blocos de $\mathcal{T}$ nós. A Figura 3.10 ilustra um exemplo desta estrutura, com $\mathcal{T} = 7$, para a Max-tree da Figura 3.7. Uma lista dinâmica de blocos (B_1, B_2, ...) é definida. Na primeira inserção, um bloco B_1 para 7 nós é alocado. Portanto, um novo bloco B_2, com mesma quantidade de espaço do primeiro, será necessário apenas na inserção do oitavo nó. Generalizando, a alocação de um novo bloco k é feita apenas para o nó de número $(k-1)\cdot\mathcal{T}+1$ na sequência de inserção. Por meio de experimentos, $\mathcal{T} = 1000$ foi adotado como uma constante por reduzir significativamente, na média, o tempo de construção da Max-tree. Novamente pode ocorrer desperdício de memória no último bloco (999 posições livres, no pior caso com apenas um nó). Se a alocação de nós fosse feita para cada pixel da imagem, tal problema poderia ser ainda maior, visto que a quantidade de componentes (nós) é normalmente muito menor que a de pixels. Uma alternativa seria usar a quantidade de zonas planas (que caracteriza um

nó) mas, ainda assim, mais de uma zona plana pode pertencer ao mesmo componente. Além disto, haveria processamento extra para a determinação desta quantidade.

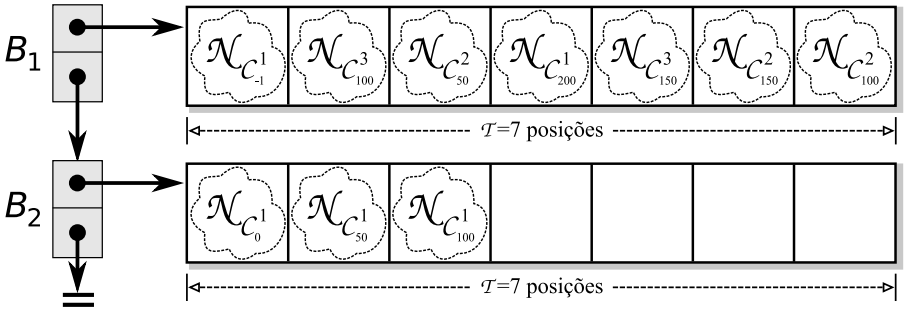


Fig. 3.10: Banco de alocação de nós.

Nó (ou nodo). Resta finalmente conhecer a descrição completa proposta e implementada para um nó. A Figura 3.11 sintetiza todas as informações armazenadas (a descrição de cada campo segue à direita, como comentário de código, após os caracteres "//") e evidencia a necessidade de uso racional de memória (são aproximadamente 80 bytes para a estrutura), além da preocupação com desempenho mencionada, para construção da árvore. Padroniza-se nível -1 e rótulo 0 para o nó-cabeça de referência à raiz (conforme linha 32 do Alg. 3.1).

3.4.3 Generalização de vizinhança

A linha 20 do Algoritmo 3.1 consiste na verificação dos vizinhos $\mathcal{V}_E(p)$ do último pixel p visitado (extraído da fila hierárquica no nível de cinza n) para decisão sobre a continuidade da inundação recursiva. Na literatura, é possível optar pelo uso de *vizinhança-4* ou *vizinhança-8* por serem naturais no sentido de definirem a conectividade de um componente de nível. Propõe-se aqui, no entanto, uma flexibilização deste parâmetro de modo que componentes desconexos por estas duas vizinhanças, possam eventualmente ser encarados como um único componente. A Figura 3.12(a) exemplifica diferentes árvores construídas para três vizinhanças definidas. A Figura 3.12(b) ilustra uma aplicação de filtragem (por altura e área), onde um único componente é criado pelo agrupamento das letras, assim como um único componente para todos os números da placa. Porém, este último, com área maior, é removido na filtragem por enxerto (vide Cap. 4).

3.4.4 Reconstrução de um componente de nível

Segundo a proposta de Max-tree apresentada, cada nó mantém uma semente de reconstrução, ou seja, uma coordenada $x = (x_{lin}, x_{col})$ de uma zona plana da imagem original I pertencente ao

```
class No_MaxTree {
    short int      nivel;        //nível  do nó
    unsigned int   rotulo;       //rótulo do nó

    No_MaxTree    *pai;          //ponteiro para o nó pai
    No_MaxTree    *filho;        //ponteiro para o nó filho mais velho
    No_MaxTree    *irmao;        //ponteiro para um irmão mais novo
    No_MaxTree    *proximo;      //ponteiro para o próximo nó da lista

    bool           ativo;        //indicador se nó foi ou não filtrado
    bool           visitado;     //auxiliar para atualização de atributos
    unsigned int   particao;     //rótulo para segmentação da árvore

    unsigned int   area;         //área   do componente
    unsigned char  altura;       //altura do componente
    unsigned int   volume;       //volume do componente

    unsigned int   x,y;          //semente de reconstrução do componente
    unsigned int   xmin,ymin;    //canto superior esquerdo da caixa envolvente do componente
    unsigned int   xmax,ymax;    //canto inferior direito  da caixa envolvente do componente
    unsigned int   xcen,ycen;    //centróide do componente

    unsigned int   grau;         //número de filhos
    unsigned int   descendentes; //número de descendentes
    unsigned char  alturatop;    //altura da sub-árvore
    unsigned char  niveltop;     //nível topológico do nó

    float          nivel_medio;  //nível médio de cinza do componente
    float          nivel_desvio; //disperção dos níveis de cinza do componente
    unsigned char  nivel_max;    //nível máximo de cinza do componente
};
```

Fig. 3.11: Descrição detalhada do nó da árvore

componente de nível associado. O Algoritmo 3.3 implementa tal reconstrução a partir de um nó $\mathcal{N}$ selecionado, acrescentando o componente $\mathcal{C}_{\mathcal{N}}$ a uma imagem I_r (que pode ser inicialmente de zeros ou de reconstruções de outros nós já processadas). Basicamente, uma inundação é feita, a partir da coordenada semente $\mathcal{N}_x$, em que, para cada pixel q visitado, o nível de cinza do nó $\mathcal{N}_{nivel}$ é comparado com o valor $I(q)$ e $I_r(q)$ (linha 5). $I_r(q)$ só é atualizado se $I(q) \geq \mathcal{N}_{nivel}$ (caso contrário, q faz parte de um nó ancestral) e $I_r(q) < \mathcal{N}_{nivel}$ (se pixel não visitado, então $I_r(q) = 0$; se já visitado em reconstruções anteriores, atualiza apenas se intensidade for superior). Na Figura 3.13, são visualizados dois componentes reconstruídos individualmente da imagem da calculadora. Esta funcionalidade é interessante quando há necessidade de reconstrução de um nó específico de maneira eficiente (requerido no algoritmo de detecção de formas a ser apresentado no Cap. 6).

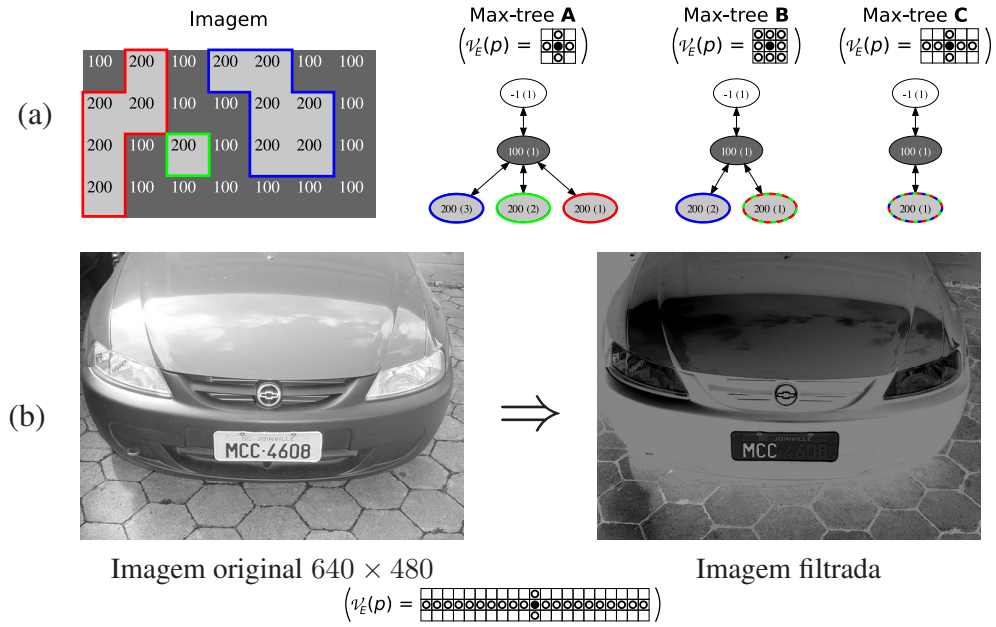


Fig. 3.12: Vizinhança genérica. (a) Árvores para três vizinhanças. (b) Exemplo de aplicação.

Fig. 3.13: Exemplos de reconstruções individualizadas de dois componentes de nível a partir de seus atributos “semente”.

Algoritmo 3.3: Algoritmo para reconstrução de um componente de nível.

ENTRADA: $\mathcal{N}$, I, $\mathcal{V}_E$, I_r
SAÍDA: I_r

RECONSTRUÇÃO_COMPONENTE()
1 $pilha$.push($\mathcal{N}_x$) *//semente*
2 **enquanto** $pilha \neq \emptyset$
3 $p \leftarrow pilha$.pop()
4 **para cada** $q \in \mathcal{V}_E(p)$
5 **se** $I[q] \geq \mathcal{N}_{nivel}$ **e** $I_r[q] < \mathcal{N}_{nivel}$
6 $pilha$.push(q)
7 $I_r[q] \leftarrow \mathcal{N}_{nivel}$
8 **retorne** I_r

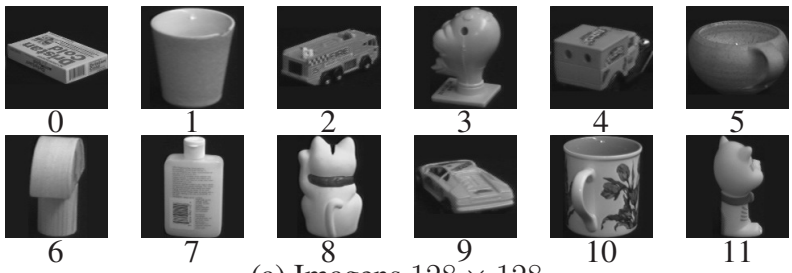
3.5 Desempenho do algoritmo

O desempenho do algoritmo estendido da Max-tree, desenvolvido neste capítulo, é analisado de modo a verificar se a eficiência de execução e uso de memória são compatíveis com outras soluções como a de [36] (**NC06**), que constrói a estrutura em tempo quase-linear, e a da recente *[831]* (**SDC08**) com o estado da arte de ferramentas morfológicas. Os testes são feitos[5] de modo a explorar situações variadas de composição estrutural de imagens em diferentes tamanhos. Todos os métodos são implementados em linguagem C++. Cabe aqui agradecimento ao Prof. Laurent Najman pela disponibilização do código-fonte (**NC06**).

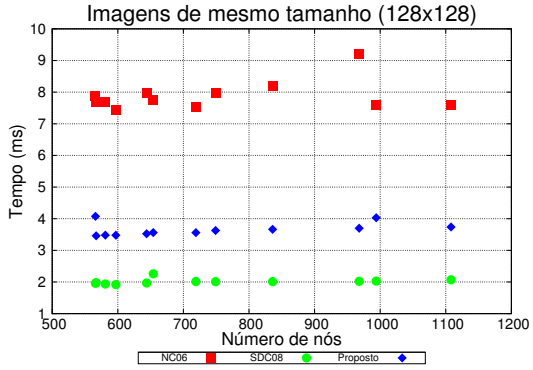
Teste A – Imagens de mesmo tamanho. As imagens dos testes da Figura 3.14, oriundas da base COIL-100 [84], têm dimensões fixas de 128×128 (16384 pixels). A quantidade de nós de suas árvores está entre 500 e 1200. O gráfico (a) ilustra o tempo de execução, em milissegundos, e o gráfico (b), o uso de memória pela Max-tree, em quilobytes, para os três algoritmos. A proposta, mesmo com introdução de uma série de novos atributos, é equiparável aos demais algoritmos para estes casos (imagens relativamente pequenas).

Teste B – Replicações de uma mesma imagem. A Figura 3.15 utiliza nova imagem da mesma base [84]. No entanto, agora são feitas replicações em ambas direções conforme as dimensões destacadas em (a). Novamente, a proposta mantém comportamento semelhante aos demais algoritmos, tanto no quesito velocidade (b), quanto no uso de memória (c). Agora torna-se evidente o efeito do número de pixels sobre o resultado. Observe que os eixos dos gráficos estão em escala logarítmica (*log-log*) para permitir melhor visualização dos pontos. A tendência de crescimento é, de fato, "quase-linear"

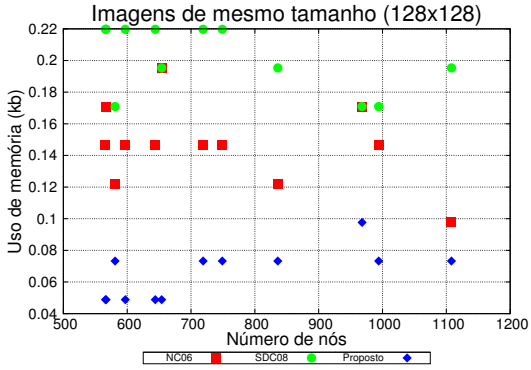
[5] Máquina utilizada nos experimentos: Pentium© Core™ 2 Duo, 2.0GHz, cache 2MB, RAM 3GB.



(a) Imagens 128×128



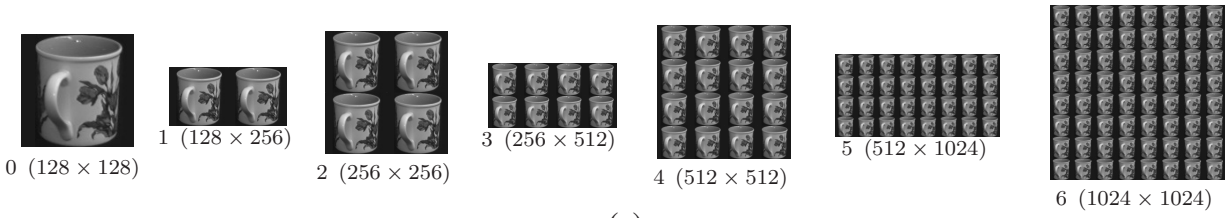
(b)



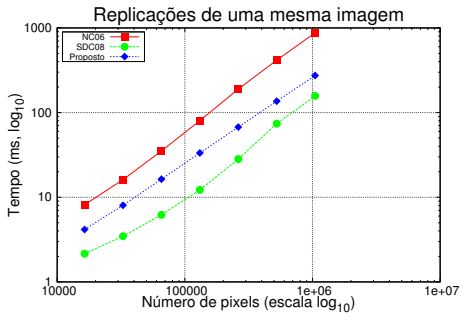
(c)

Fig. 3.14: Desempenho de construção da árvore para imagens de mesmo tamanho. (a) Imagens de teste. (b) Tempo em função do número de nós. (c) Memória utilizada em função do número de nós.

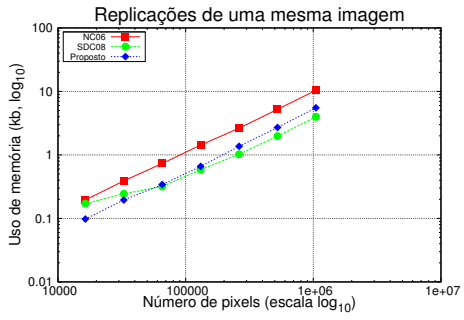
para este conjunto.



(a)



(b)



(c)

Fig. 3.15: Desempenho de construção da árvore para replicações de uma mesma imagem. (a) Imagens de teste (fora de escala). (b) Tempo de execução. (c) Memória utilizada.

Teste C – Mesma imagem em vários tamanhos. A Figura 3.16 ilustra o tempo para uma mesma imagem de um lago, originalmente de 1280×960, reduzida ambas dimensões à metade 4 vezes (a). Um novo gráfico (b) é apresentado com a relação praticamente linear – com coeficiente angular abaixo de 1 como esperado (número de nós muito menor que o número de pixels) – entre as quantidades de nós e de pixels (b). Os tempos, em (c), e memória utilizada, em (d), também estão próximos de outras soluções da literatura. Interessante notar que a proposta se destaca pela previsibilidade linear de duração e uso de recursos no caso de escala de uma única imagem.

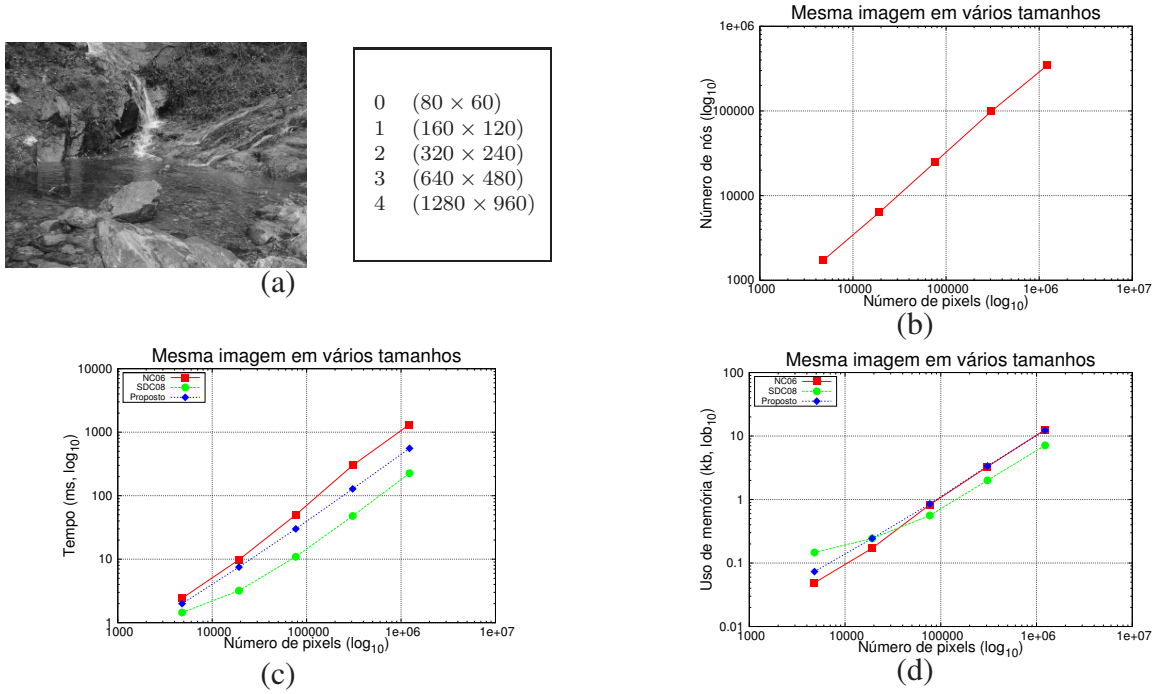


Fig. 3.16: Desempenho de construção da árvore para uma mesma imagem em vários tamanhos. (a) Imagem de teste. (b) Número de nós em função do número de pixels. (c) Tempo de execução. (d) Memória utilizada.

Teste D – Imagens aleatórias. Por fim, quinze imagens aleatórias foram geradas, de 100×100 até 1500×1500, conforme indicado na Figura 3.17(a). O número de nós, neste caso, é relativamente alto em relação ao número de pixels (b) (há $1,8$ vezes menos nós que pixels em todos os testes). Para este conjunto de imagens sintéticas, a proposta continua com tempo de processamento intermediário ao das demais propostas, mas passa a consumir relativamente mais memória em função da quantidade elevada de nós (lembrando que há uma série de novos atributos sendo associada a cada nó).

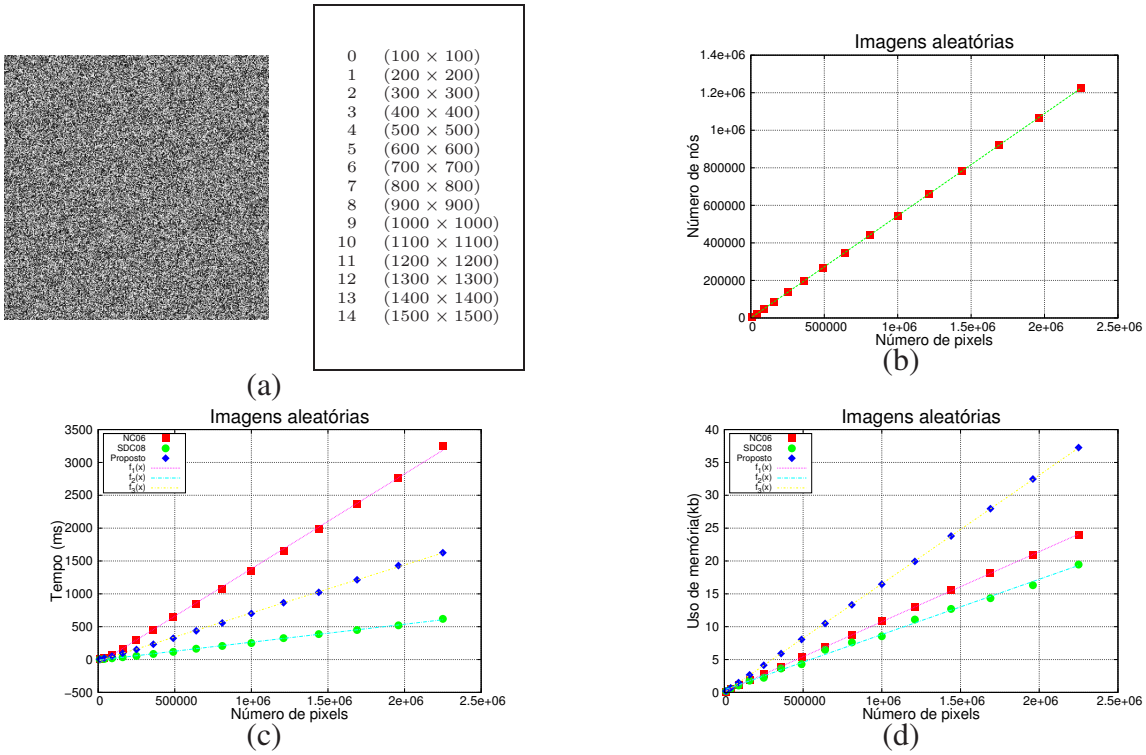


Fig. 3.17: Desempenho de construção da árvore para imagens aleatórias. (a) Imagens de teste. (b) Número de nós em função do número de pixels. (c) Tempo de execução. (d) Memória utilizada.

3.6 Considerações sobre o capítulo

A estrutura de dados da Max-tree é estendida para suportar um conjunto de novos atributos, importantes no desenvolvimento de operadores diferenciados (Cap. 4), no estabelecimento de valores originais de extinção (Cap. 5) ou detecção genérica de formas (Cap. 6). Destaca-se que o algoritmo de [36] determina **área**, **altura** e **volume**, enquanto o algoritmo de [83] acrescenta as coordenadas de **caixa envolvente**, no processo de construção da árvore. A proposta neste capítulo, além destes, determina **centroide**, **nível topológico**, **altura de subárvore**, **grau do nó**, **número de descendentes**, **semente de reconstrução**, além de **nível médio de cinza**, **desvio padrão**, **máximo** e **mínimo** de intensidades, para cada nó (componente de nível). Mesmo com o acréscimo de tais atributos, se mostrou eficiente nos casos de teste, em comparação às demais soluções, tanto em relação a tempo de processamento como uso de memória, a partir de implementação de uma arquitetura predominantemente estática de memória e buscas por meio de *hashings*. Ressalta-se que todos os atributos foram calculados em uma única varredura da imagem. Os mesmos poderiam ser determinados, para cada componente conexo, a partir das L imagens obtidas pela decomposição por limiares. No entanto, para isso, haveria necessidade de uma etapa de rotulação para cada uma dessas imagens, isolamento

de cada componente conexo rotulado e cálculo de seus atributos. Além disto, não haveria um procedimento direto para determinação dos atributos topológicos.

A Tabela 3.1 mostra as características principais das implementações, como complexidade, atributos e tempo médio de execução para 49 imagens[6] de 512×512 (observa-se, para este conjunto de testes em particular, a proposta, que determina uma quantidade maior de atributos, mais de duas vezes mais rápida que **NC06** e mais de duas vezes mais lenta que **SDC08**). A razão de a **SDC08** ter um desempenho melhor está fundamentalmente na forma enxuta de representação da árvore em memória, apenas com opções essenciais de navegação na topologia, e limitação do número de atributos, aspecto este que dificulta o desenvolvimento de filtros baseados em topologia por exemplo. Uma vez apresentados todos os detalhes relativos à árvore de componentes ou Max-tree, passa-se agora à etapa de desenvolvimento de ferramentas, baseadas nesta estrutura, para filtragem, segmentação e reconhecimento. Os próximos capítulos são elaborados com este intuito.

Tab. 3.1: Resumo das características principais das implementações de Max-tree.

	NC06	**SDC08**	**Proposto**
Algoritmo	baseado em *union-find* [36]	baseado em *flooding* [1]	baseado em *flooding* [1] e estruturas de *hashing*
Complexidade	$O(n \times \alpha(n))$, $\alpha(n) < 4$	$O(L \times n)$	$O(L \times n)$
Atributos	área, altura, volume, intensidade mínima	área, altura, volume, caixa envolvente, intensidade mínima	área, altura, volume, caixa envolvente, centroide, nível topológico, altura de subárvore, grau do nó, número de descendentes, sementre de reconstrução, nível médio de cinza, desvio padrão, intensidades mínima e máxima
Testes (ms)	T_{nc} $218,13 \pm 25,23$ $(5,70T_{sdc}; 2,34T_{prop})$	T_{sdc} $38,33 \pm 10,54$ $(0,18T_{nc}; 0,41T_{prop})$	T_{prop} $93,14 \pm 21,20$ $(0,43T_{nc}; 2,43T_{sdc})$

[6] `http://decsai.ugr.es/cvg/CG/base.htm`

Capítulo 4

Operadores conexos antiextensivos

Uma imagem é processada, em grande parte das aplicações de visão computacional, a partir do tratamento global ou local de pontos. No entanto, a informação semântica é limitada se os elementos da representação forem puramente pontos (com informação de cor, por exemplo), sobretudo no que se refere à identificação, caracterização e interpretação dos objetos (regiões) constituintes. Neste sentido, sugere-se nova forma de representação da imagem para que dois objetivos principais possam ser alcançados: **(i)** melhoramento dos algoritmos de processamento de imagens existentes, tornando suas execuções mais rápidas; **(ii)** desenvolvimento de novos algoritmos com ampla possibilidade de parametrização. Neste capítulo, são apresentados alguns operadores conexos e antiextensivos (Def. 2.15) derivados do relacionamento de regiões da imagem. Alguns destes – como as filtragens por área (abertura por área), altura (ou profundidade) e volume – conhecidos, outros conhecidos mas com implementação mais eficiente a partir da estrutura de dados em estudo, e os demais sendo propostas de algoritmos que exploram outros atributos e topologia da árvore. O processamento de árvores de componentes, como alternativa ou complemento a técnicas no domínio da imagem, se configura como tópico recorrente na literatura, em especial sob denominação de filtros/abertura por reconstrução [85, 86], filtros/abertura de/por atributos [41, 15, 87, 19], afinamento/espessamento por atributos [10, 16], entre outras, e seu uso está em constante aprimoramento em processamento de imagens, como observado na cronologia de algumas publicações em conferências [15, 87, 19, 32], periódicos [10, 1, 41, 32] ou teses [88, 11, 89]. Porém o tema não se esgota e as possibilidades de manipulação destas estruturas, com o propósito de gerarem ferramentas inovadoras, é o foco deste capítulo.

4.1 Filtragem

A seleção de nós de uma Max-tree MT_I, baseada na observação de seus atributos e informação topológica disponível, é importante na redução do número de componentes de I, no sentido de torná-la mais apropriada para segmentação, reconhecimento ou casamento de imagens. Em outras palavras, um pré-processamento pode ser feito eficientemente. A filtragem discutida tem caráter mais amplo que abertura morfológica [90, 91], abertura por atributo [92, 10] ou filtro por atributo (vide Def. 2.17 do Cap. 2). De forma geral, um filtro morfológico deve ser crescente e idempotente [93, 94]. Um filtro por atributo é crescente e antiextensivo. Já os filtros desenvolvidos sobre a Max-tree tem obrigatoriedade apenas da antiextensividade, podendo eventualmente se especializar nos filtros anteriores. Os filtros apresentados também podem ser classificados quanto à forma de remoção de componentes da imagem, sendo denominados, neste trabalho, de *poda*[1] e *enxerto*[2] em analogia a ações semelhantes em árvores naturais.

Definição 4.1 (Poda) *A* poda $\psi_{\mathcal{C}^R}^{poda}$ *consiste na preservação de componentes diferentes e não contidos em* $\mathcal{C}^R$ *ou da remoção de uma subárvore completa da Max-tree enraizada no nó* $\mathcal{N}^R$*:*

$$\begin{array}{rrcl} \textit{Domínio da imagem:} & \psi_{\mathcal{C}^R}^{poda}(I)(p) & = & \max_{(\forall n,\forall r,\forall p\in E)}\{\mathscr{C}_n^r(I)(p) \mid \mathcal{C}_n^r(I) \nsubseteq \mathcal{C}^R\} \\ \textit{Domínio da árvore:} & \psi_{\mathcal{N}^R}^{poda}(MT_I) & = & \{\forall \mathcal{N} \in MT_I \mid \mathcal{N} \neq \mathcal{N}^R \textbf{ e } \mathcal{N} \notin \mathcal{N}_{DESC}^R\} \end{array} \tag{4.1}$$

sendo $\mathcal{N}_{DESC}^R$ *o conjunto de nós descendentes de* $\mathcal{N}^R$*.*

Definição 4.2 (Enxerto) *O* enxerto $\psi_{\mathcal{C}^X}^{enxerto}$ *consiste na remoção de um único componente qualquer* $\mathcal{C}^X$ *ou preservação de todos os nós, exceto* $\mathcal{N}^X$*. Caso o componente seja um máximo regional (*$\mathcal{C}^X \cap Maxreg(I) = \mathcal{C}^X$*) ou, de outra forma, caso o nó seja uma folha (*$\mathcal{N}_{grau}^X = 0$*), então tem-se poda como definição mais apropriada.*

$$\begin{array}{rrcl} \textit{Domínio da imagem:} & \psi_{\mathcal{C}^X}^{enxerto}(I)(p) & = & \max_{(\forall n,\forall r,\forall p\in E)}\{\mathscr{C}_n^r(I)(p) \mid \mathcal{C}_n^r(I) \neq \mathcal{C}^X)\} \\ \textit{Domínio da árvore:} & \psi_{\mathcal{N}^X}^{enxerto}(MT_I) & = & \{\forall \mathcal{N} \in MT_I \mid \mathcal{N} \neq \mathcal{N}^X\} \end{array} \tag{4.2}$$

A Figura 4.1 sintetiza estas duas visões de filtragem. A poda da Max-tree tem como efeito a remoção de elevações na imagem. Reconstrução morfológica [45], filtros por reconstrução [44, 96], abertura por área [92] são exemplos clássicos de poda. Já o nome enxerto é sugerido aqui por representar a inserção de novos componentes no local de remoção do nó, conforme o preenchimento

[1] *Pruning* é bastante utilizado em outros contextos, como o da transformada imagem-floresta [95], mas também é uma operação mencionada sobre a Max-tree [31, 28]. *Min* ou *max decision* [1] são, de fato, denominações encontradas com sentido de poda da Max-tree.

[2] *Grafting* não é um termo usual e diz-se simplesmente filtro baseado em critério não crescente [1, 5] ou por *non-pruning strategies* [31]. Se critério for um atributo (e não uma relação topológica), *direct decision* [1] é um termo utilizado com a mesma funcionalidade do enxerto.

feito (em azul) na Figura 4.1 (árvore abaixo e à direita). Um conjunto de enxertos pode ser efetuado pela filtragem da Max-tree por meio de seleção de atributos não crescentes. Alguns exemplos serão mostrados ao longo deste capítulo.

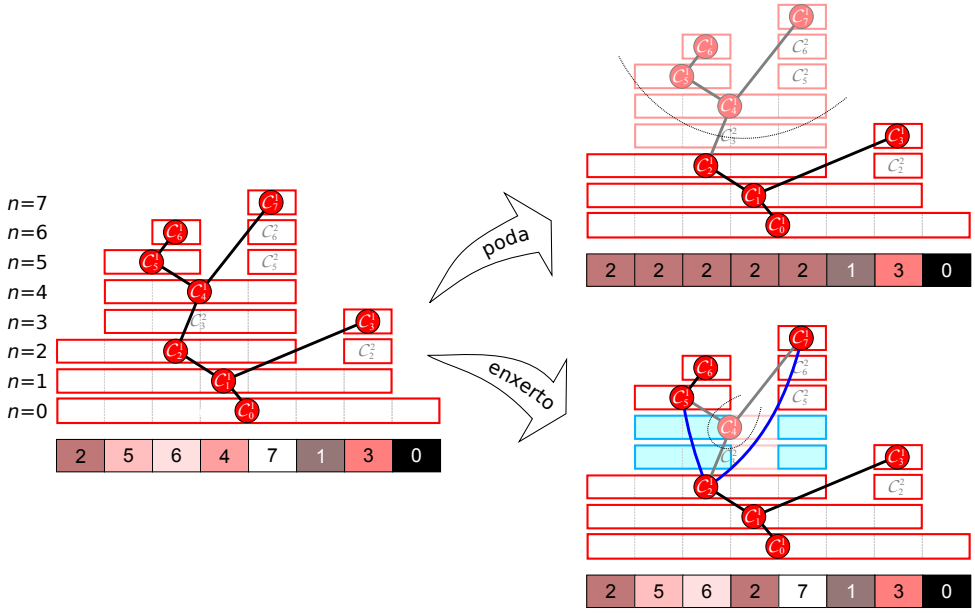


Fig. 4.1: Filtragem por poda e enxerto do componente $\mathcal{C}_4^1$.

A Figura 4.2 ilustra o esquema geral de filtragens da árvore, seja de poda e/ou enxerto. Inicialmente constrói-se a Max-tree MT_E, a partir de uma imagem E de entrada, pelo algoritmo apresentado no Capítulo 3. A partir deste ponto, a filtragem pode ser baseada nos atributos definidos em cada nó, com possibilidade de vários deles (filtragem 1 a k) serem analisados sob uma mesma condição, conforme o Algoritmo 4.1. Cada filtragem, de 1 a k, também pode ser baseada nas relações topológicas entre os nós. Alguns filtros, com este perfil, serão propostos neste capítulo. Feita esta primeira etapa de podas e/ou enxertos, obtém-se uma nova árvore com atributos atualizados (linha 4 do Alg. 4.1) – alguns atributos como área do componente, por exemplo, não se altera após uma filtragem, porém a maioria deles, como altura, volume, grau, descendentes do nó, nível e altura topológica, são modificados e devem ser atualizados. Esta árvore intermediária MT_I pode sofrer novas filtragens e assim sucessivamente até a obtenção da Max-tree final de saída MT_S, cuja reconstrução gera a imagem filtrada resultante S. Veja que todo o processamento de filtragem é feito no domínio da árvore, possibilitando implementações significativamente mais rápidas.

O Algoritmo 4.1 implementa a filtragem da Max-tree baseada na comparação (um atributo μ está ou não em um intervalo $[d_{min}, d_{max}]$) dos atributos existentes $(\mu_1, \mu_2, \ldots, \mu_k)$ em cada nó. Em um mesmo percurso na árvore, pode-se analisar diversos atributos e aninhar as comparações de formas variadas com "**e**" e "**ou**" lógicos, o que pode significar, ao final do processo, um ganho de desempenho

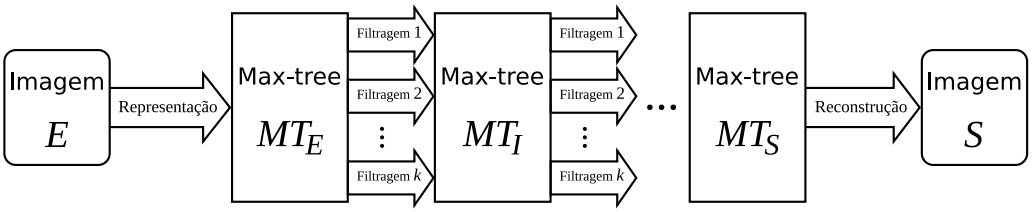


Fig. 4.2: Esquema geral de filtragens usando a Max-tree.

em relação a soluções idênticas feitas com percursos consecutivos na árvore ou por meio de filtragens consecutivas análogas no domínio da imagem. A remoção de nó (linha 3 do Alg. 4.1) consiste simplesmente em marcá-lo como inativo, de modo que possa ser utilizado auxiliarmente no processo de reconstrução (vide Alg. 4.3 a seguir) da imagem S de saída. Resta agora entender o procedimento adotado para atualização de atributos (incluindo aqueles introduzidos no Cap. 3), quando necessário, assim como a reconstrução para o mapeamento de nós, remanescentes da árvore, em componentes de nível da imagem filtrada.

Algoritmo 4.1: Algoritmo para filtragem de poda e enxerto da Max-tree baseada em seus atributos.

FILTRA_MAX-TREE_ATRIBUTOS()
1 **para cada** $\mathcal{N} \in MT_I$
2 **se** $d_{1_{min}} \leq \mathcal{N}_{\mu_1} \leq d_{1_{max}}$ **e** | **ou** $d_{2_{min}} \leq \mathcal{N}_{\mu_2} \leq d_{2_{max}}$ **e** | **ou** $\cdots d_{k_{min}} \leq \mathcal{N}_{\mu_k} \leq d_{k_{max}}$
3 MT_I.remove($\mathcal{N}$) *//marcar $\mathcal{N}$ como inativo*
4 ATUALIZA_ATRIBUTOS(MT_I)

4.1.1 Atualização de atributos

Alguns atributos são modificados após uma filtragem. No domínio da árvore, grau, número de descendentes, altura topológica e nível topológico de um nó obviamente se alteram com podas ou enxertos. No domínio da imagem, altura e volume de parte dos nós também passam a assumir novos valores com a remoção associada de componentes de nível. O Algoritmo 4.2 é responsável pela atualização dos atributos de cada nó da Max-tree caso ao menos um nó seja excluído da árvore. A linhas $1-6$ consistem na atualização do nível topológico de cada nó por meio de um percurso em largura (tempo linear). A inicialização (atribuição de zero) dos demais atributos, no domínio da árvore, também é feita neste primeiro passo. A partir deste ponto, de cada uma das folhas (linha 7), caminha-se em direção à raiz (linha 11), e verifica-se se o nó (filho) já foi visitado (linha 12) para efetuar ou não a atualização de volume e número de descendentes (linha 13), e grau (linha 14). Por fim, ajusta-se o cálculo de volume e número de descendentes (linha 15). A altura e altura topológica são atualizadas apenas no caso de o máximo dos filhos, para estes mesmos atributos, tenha sido alterado com a filtragem (linhas $16-17$). Exceto o nível topológico, que é definido isoladamente no

percurso inicial, em largura, os demais atributos topológicos seguem a mesma ideia de, a partir das folhas, caminhar até a raiz, verificando nós visitados e ramificações (nós com filhos) da árvore. A Figura 4.3 exemplifica a atualização destes atributos, para cada caminho (arestas em vermelho), de cada uma das quatro folhas ($\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$ e $\mathcal{N}_4$). Os nós já visitados são marcados (preenchimento em amarelo). O grau de um nó ($\mathcal{N}$) é incrementado se, no caminho, seu filho ($\mathcal{N}^F$) ainda não tiver sido visitado (Fig. 4.3a). Ao número de descendentes de um nó, é somado o número de descendentes de seu filho mais um, caso este ainda não tenha sido visitado (Fig. 4.3b). Quanto à altura topológica, atribui-se a do filho, se este ainda não visitado, e soma-se um, apenas quando tal quantia implicar em um valor maior que sua altura topológica atual (Fig. 4.3c). Evidentemente estes três atributos são atualizados de uma só vez, ou seja, apenas um percurso, a partir de cada folha, é necessário.

Algoritmo 4.2: Algoritmo para atualizar os atributos de cada nó da Max-tree após uma filtragem.

ATUALIZA_ATRIBUTOS(MT_I)
1 $\mathcal{N} \leftarrow$ raiz de MT_I; $\mathcal{N}_{ntop} \leftarrow 0$; $fila$.insere($\mathcal{N}$); $altura \leftarrow 0$
2 **enquanto** $fila \neq \emptyset$ *//percurso em largura para atualizar ntop e zerar atributos*
3 $\mathcal{N} \leftarrow fila$.remove()
4 $\mathcal{N}_{grau} \leftarrow 0$; $\mathcal{N}_{desc} \leftarrow 0$; $\mathcal{N}_{htop} \leftarrow 0$; $ntop \leftarrow \mathcal{N}_{ntop} + 1$
5 marcar $\mathcal{N}$ como não visitado; $\mathcal{N} \leftarrow$ filho de $\mathcal{N}$
6 **enquanto** $\exists\mathcal{N} \implies (\mathcal{N}_{ntop} \leftarrow ntop$; $fila$.insere($\mathcal{N}$); $\mathcal{N} \leftarrow$ irmão de $\mathcal{N})$
7 **para cada** $\mathcal{N} \in$ (folhas de MT_I) *//atualiza altura, volume, grau, htop e desc*
8 $\mathcal{N}_{altura} \leftarrow 0$; $\mathcal{N}_{volume} \leftarrow 0$; $\mathcal{N}_{desc} \leftarrow 0$; $\mathcal{N}_{htop} \leftarrow 0$
9 $desc \leftarrow 0$; $htop \leftarrow 0$; $volume \leftarrow 0$
10 $alturabase \leftarrow \mathcal{N}_{nivel}$; $\mathcal{N}^F \leftarrow \mathcal{N}$; $\mathcal{N} \leftarrow$ pai de $\mathcal{N}$
11 **enquanto** $\exists\mathcal{N}$
12 **se** $\mathcal{N}^F$ **não** visitado
13 $volume \leftarrow volume + \mathcal{N}^F_{area} \cdot \left(\mathcal{N}^F_{nivel} - \mathcal{N}_{nivel}\right)$; $desc \leftarrow desc + 1$
14 $\mathcal{N}_{grau} \leftarrow \mathcal{N}_{grau} + 1$; marcar $\mathcal{N}^F$ como visitado
15 $\mathcal{N}_{volume} \leftarrow \mathcal{N}_{volume} + volume$; $\mathcal{N}_{desc} \leftarrow \mathcal{N}_{desc} + desc$; $htop \leftarrow htop + 1$
16 **se** $\mathcal{N}_{altura} < altura \implies (altura = alturabase - \mathcal{N}_{nivel}$; $\mathcal{N}_{altura} \leftarrow altura)$
17 **se** $\mathcal{N}_{htop} \leq htop \implies \mathcal{N}_{htop} = htop$
18 $\mathcal{N}^F \leftarrow \mathcal{N}$; $\mathcal{N} \leftarrow$ pai de $\mathcal{N}$

4.1.2 Reconstrução da imagem

Como visto, filtragens são feitas exclusivamente no domínio da árvore. Porém, pretende-se produzir uma imagem resultante de podas ou enxertos. Uma forma eficiente de mapear (renderizar) o subconjunto de nós da Max-tree para o domínio da imagem é apresentada no Algoritmo 4.3. Utiliza-se, como referência, a imagem I original. Para cada um de seus pixels (linha 2), a partir da intensidade e rótulo na matriz $status$, criada no processo de construção da Max-tree, verifica-se o nó associado

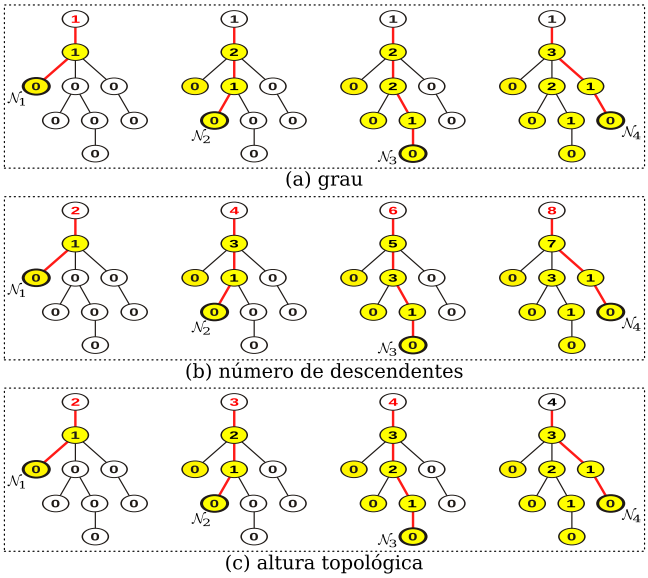


Fig. 4.3: Ilustração da atualização de atributos topológicos.

pela estrutura *hashing* (linha 3). Se este está ativo (linha 4), ou seja, se não foi removido no processo de filtragem (Alg. 4.1), então o nível de cinza da imagem resultante replica o da imagem de entrada (linha 5). Caso contrário (linha 6), busca-se o pai ativo deste nó, ou seja, busca-se a intensidade de um componente que persiste após a filtragem (linha 7). Caso não se trate da raiz, que não possui pai (linha 8), atribui-se então o nível deste nó para o pixel da imagem de reconstrução (linha 9). Este procedimento consiste na reconstrução de todos os componentes remanescentes. Eventualmente pode-se reconstruir, destes nós, apenas aqueles marcados por algum mecanismo de seleção (no Cap. 5, é exibido um destes mecanismos, no caso, de segmentação de imagens pela rotulação de subárvores).

Algoritmo 4.3: Algoritmo para reconstrução (renderização) da imagem a partir da árvore filtrada.

ENTRADA: MT_I
SAÍDA: I_R

RECONSTRUÇÃO_IMAGEM()
1 $I_R[x] \leftarrow 0, \forall x \in E \subset \mathbb{N}^2$
2 **para cada** $p \in E$ (varredura *raster* na imagem I)
3 $\quad \mathcal{N} \leftarrow MT_I$.HASH_TABLE($\{I[p], status[p]\}$)
4 $\quad$ **se** $\mathcal{N}$ está ativo (não filtrado/removido)
5 $\quad\quad I_R[p] = I[p]$
6 $\quad$ **senão**
7 $\quad\quad \mathcal{N}^P \leftarrow$ pai (ativo) de $\mathcal{N}$
8 $\quad\quad$ **se** $\exists \mathcal{N}^P$
9 $\quad\quad\quad I_R[p] \leftarrow \mathcal{N}^P_{nivel}$
10 **retorne** I_R

4.1.3 Filtros topográficos

Considera-se, neste trabalho, filtragem topográfica como sendo aquela que age em atributos intimamente relacionados com a representação do relevo ou superfície da imagem. Remete primordialmente, portanto, à ideia de processamento de características de componentes no domínio da imagem. Desta forma, suas implementações não exigem necessariamente a utilização de informações de uma Max-tree.

Área, altura e volume. A Figura 4.4 ilustra a filtragem de atributos de área (b), altura (c) e volume (d) para uma imagem sintética (a). Observa-se a exclusão de topos de relevo diferenciados em cada caso. Tais filtragens são comuns na literatura quando se configuram como poda, ou seja, quando há exclusão de nós cujos atributos sejam maiores ou iguais a um certo valor. Aqui, adicionalmente, também é possível estabelecer um intervalo de valores de atributos (conforme o Alg. 4.1), procedendo enxertos se necessário.

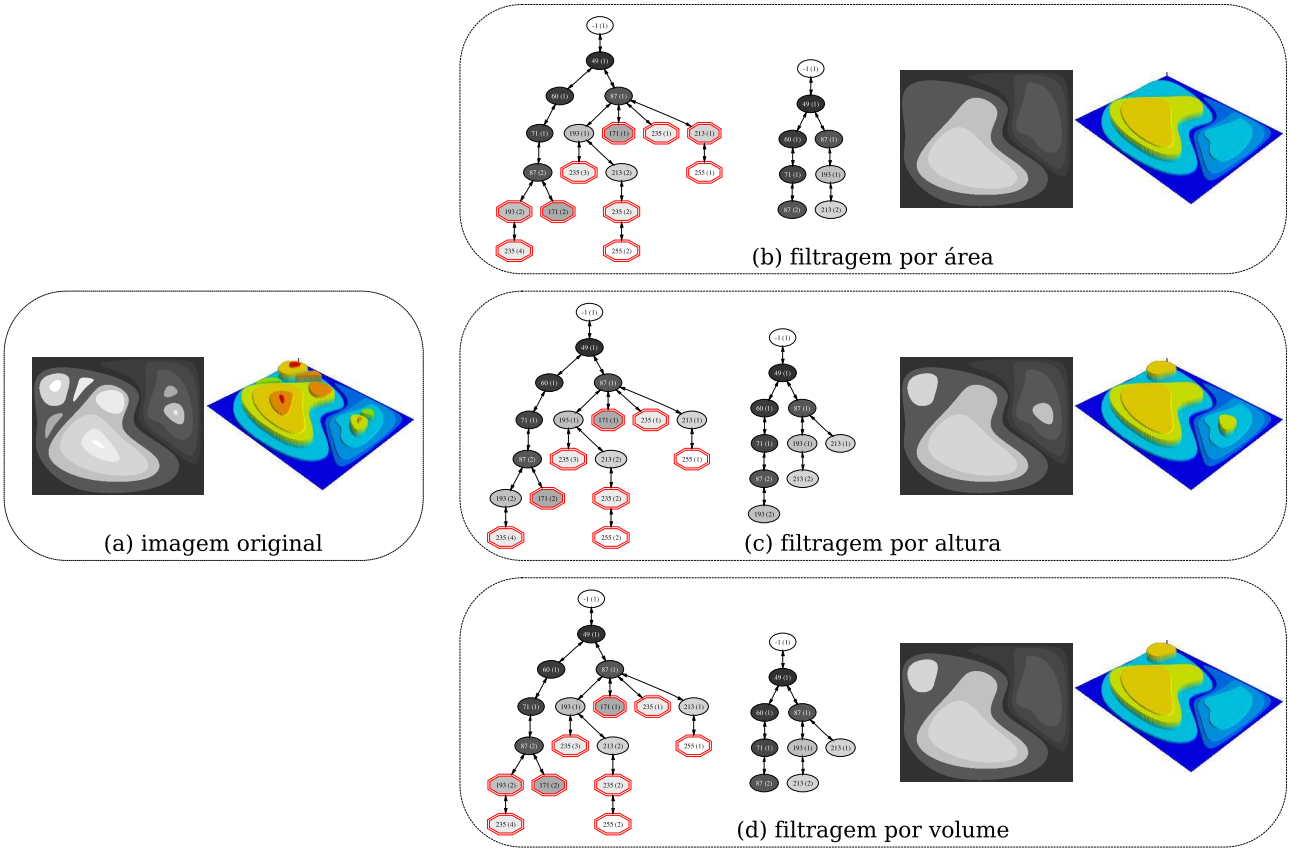


Fig. 4.4: Filtragem topográfica. (a) Imagem original 322×402. (b) Remoção de área entre 0 a 5000. (c) Remoção de altura entre 0 a 20. (d) Remoção de volume entre 0 a 10000.

4.1.4 Propostas de filtros topológicos

A filtragem topológica provém da análise exclusiva da hierarquia de nós, sendo feita no domínio da árvore (embora com interpretações na imagem). Atributos propostos neste trabalho, como nível topológico, grau, número de descendentes e altura topológica, podem ser observados e nós são descartados conforme o intervalo de valores desejado. Nesta modalidade, filtros diferenciados como, por exemplo, a eliminação de nós que não possuem irmãos, podem ser projetados para simplificação de imagens. Vale recordar que tais atributos são calculados incrementalmente, não alterando a complexidade de construção da árvore.

Nível topológico, grau e número de descendentes. Assim como os nós foram preservados ou descartados, à pouco, em função da área, altura e volume dos componentes da imagem, qualquer outro atributo pode ser verificado em filtragens (Alg. 4.1). Neste ponto, propõe-se a exclusão de nós conforme seu nível topológico, grau ou quantidade de descendentes. A Figura 4.5 ilustra esta ideia por meio de um exemplo sintético (a). A filtragem de nós pelo nível topológico (b) consiste na remoção de componentes da imagem cuja propriedade comum é a distância que têm em relação ao menor nível de cinza da imagem (raiz da árvore). A remoção de nó baseado em seu grau (número de filhos) (c), consiste na retirada de componentes que contêm certo número de outros componentes [30]. Pode-se pensar também no contrário, preservando apenas componentes dos quais surgem um certo número de elevações no relevo da imagem. Por fim, a decisão sobre a permanência de nós pode, opcionalmente, ser feita com base em seu número de descendentes (d). Um componente que contém quantidade significativa de outros componentes internos – que, por sua vez, contêm outros, e assim sucessivamente – apresenta um valor considerável de descendentes (sempre maior ou igual à quantidade de filhos). Estas simplificações podem ser interessantes no sentido de facilitar etapas de processamento subsequentes, sobretudo em imagens com constituição estrutural previsível adquiridas em ambientes controlados de iluminação.

Altura topológica. Dentre os novos atributos, também pode-se observar a altura topológica (ou altura da subárvore) associada a cada nó. Um filtro baseado na remoção de nós por imposição de limites a este atributo (Alg. 4.1) tem efeito de retirar componentes da imagem, cujas elevações internas, apresentam certa quantidade de componentes sobrepostos. A Figura 4.6(b) exemplifica esta ideia aplicada à imagem (a) pela exclusão de nós com alturas de subárvores no intervalo de 1 a 3.

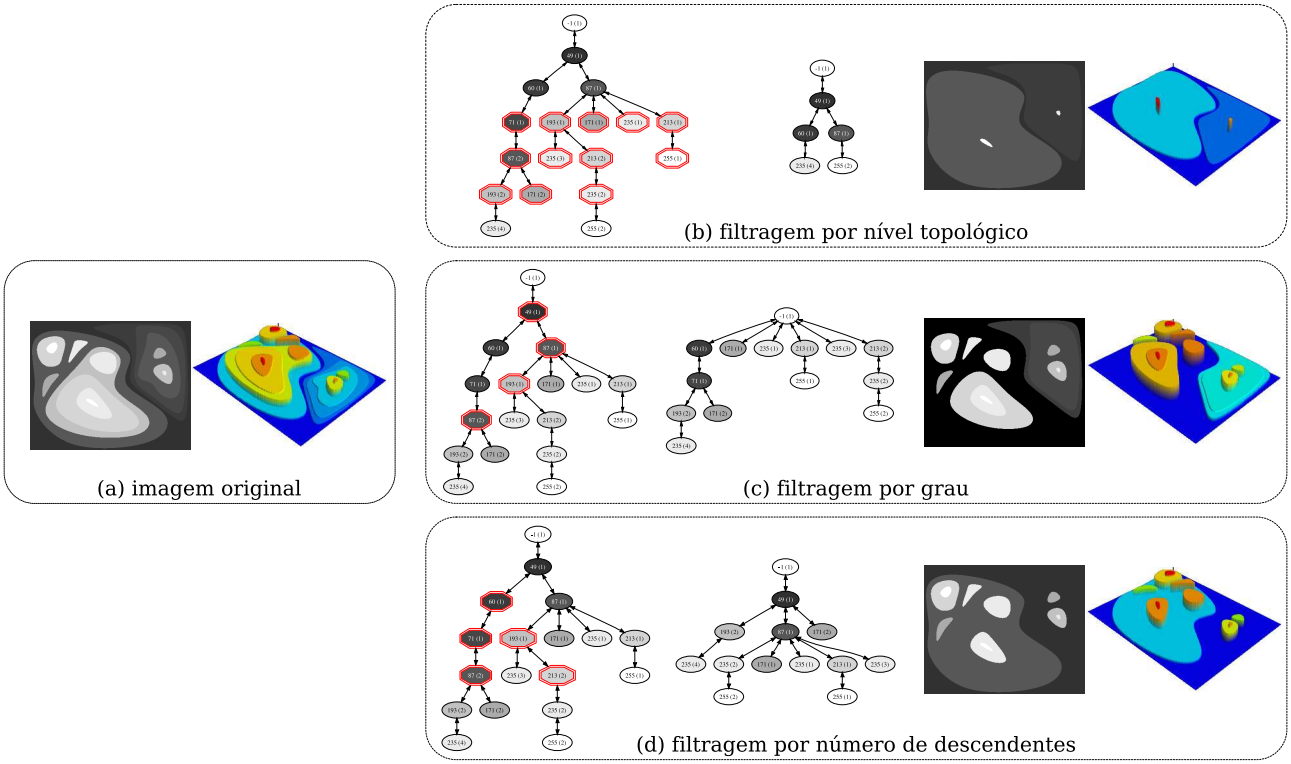


Fig. 4.5: Filtragem topológica. (a) Imagem original. (b) Remoção de níveis topológicos de 2 a 4. (c) Remoção de nós com grau de 2 a 4. (d) Remoção de nós com número de descendentes de 2 a 5.

4.1.5 Nós salientes e sem irmãos

Além da redução de nós e consequente simplificação da árvore, é possível manter a organização estrutural da imagem, no sentido de preservar todas as ramificações existentes. Para isto, mais dois filtros topológicos são desenvolvidos:

Nós salientes. Onde são removidos nós (componentes), a partir de todas as folhas (máximos regionais), até a ocorrência de uma ramificação, ou mais precisamente, até encontrar um nó que tenha ao menos mais um irmão (sendo este preservado) [30]. O Algoritmo 4.4 sintetiza esta ação e a Figura 4.6(c) mostra um exemplo desta filtragem. A Figura 4.7(a) mostra, por sua vez, o efeito da aplicação deste filtro sobre uma imagem real. Para a imagem do câmera, há uma redução de nós de um total de 15811 (imagem original) para 12114, ou seja, de aproximadamente 23%. A similaridade estrutural SSIM [97], entre as imagens, neste caso, é cerca de $0,99$. Em outras palavras, a remoção de "nós salientes" promove uma simplificação da árvore (da imagem) com manutenção do aspecto visual.

Nós sem irmãos. Onde é descartado um nó (componente), sempre que seu pai (componente pai) o envolver exclusivamente, ou seja, é feita a remoção de nós caracterizados como filho único [30].

Algoritmo 4.4: Algoritmo para filtragem de nós salientes.

FILTRA_SALIENTES()
1 **para cada** $\mathcal{N} \in$ (folhas de MT_I)
2 $\quad \mathcal{N}^P \leftarrow$ pai de $\mathcal{N}$
3 $\quad$ **enquanto** $\exists \mathcal{N}^P$ **e** $\mathcal{N}^P_{grau} = 1 \quad \Rightarrow \quad (MT.\text{remove}(\mathcal{N}); \quad \mathcal{N} \leftarrow \mathcal{N}^P; \quad \mathcal{N}^P \leftarrow$ pai de $\mathcal{N})$
4 ATUALIZA_ATRIBUTOS(MT_I)

O Algoritmo 4.5 implementa esta ideia e seu efeito pode ser visualizado na Figura 4.6(d). A árvore resultante da filtragem "sem irmãos", no negativo da imagem (ou Min-tree), assemelha-se à árvore de lagos críticos [14, 98], porém é n-ária (um nó pode ter n filhos), enquanto esta última é binária (um nó pode ter, no máximo, dois filhos). Além disto, a fusão de regiões (ou junção de componentes em um único pai) continua seguindo a relação de inclusão de componentes na Max-tree, enquanto, na árvore de lagos críticos, outros critérios, como altura (profundidade) ou volume, são adotados. Menciona-se ainda o fato de a Max-tree se restringir a operações conexas antiextensivas (o *watershed* utilizado de forma iterativa na árvore de lagos críticos, por outro lado, decide pela divisão de zonas de empate fugindo aos domínios de borda dos componentes). Figura 4.7(b) mostra a ação do filtro sobre uma imagem real. Há redução de 45% de nós e similaridade estrutural SSIM [97] de $0,59$ entre imagens filtrada e original. Em outras palavras, a remoção de "nós sem irmãos" implica em uma simplificação expressiva da árvore (da imagem), mas compromete o aspecto visual.

Algoritmo 4.5: Algoritmo para filtragem de nós sem irmãos.

FILTRA_SEM_IRMÃOS()
1 **para cada** $\mathcal{N} \in MT_I$
2 $\quad \mathcal{N}^P \leftarrow$ pai de $\mathcal{N}$
3 $\quad$ **se** $\exists \mathcal{N}^P$ **e** $\mathcal{N}^P_{grau} = 1 \quad \Rightarrow \quad MT.\text{remove}(\mathcal{N})$
4 ATUALIZA_ATRIBUTOS(MT_I)

4.1.6 Filtros estatísticos

Com relação aos atributos estatísticos de cada nó $\mathcal{N}$ da árvore, derivam-se algumas propriedades: **(i) escala local** como a diferença entre $\mathcal{N}_{n_{max}}$ e $\mathcal{N}_{n_{min}}$ – caso tal diferença seja 0 (ou se $\mathcal{N}_{n_\sigma} = 0$), trata-se de um máximo regional; caso a diferença seja máxima, trata-se do componente com mais alto intervalo de níveis de cinza (raiz) –; **(ii) homogeneidade local** que, baseada na informação de desvio padrão $\mathcal{N}_{n_\sigma}$, indica o quanto há de variação nos níveis de cinza do componente ou, de certa forma, o quanto o nível médio $\mathcal{N}_{n_\mu}$ se distancia dos extremos $\mathcal{N}_{n_{min}}$ e $\mathcal{N}_{n_{max}}$. A seleção de nós, baseada na especificação de um intervalo $[d_{min}, d_{max}]$ sobre atributos estatísticos (no Alg. 4.1), possibilita implementadação de novos operadores conexos antiextensivos. Destacam-se:

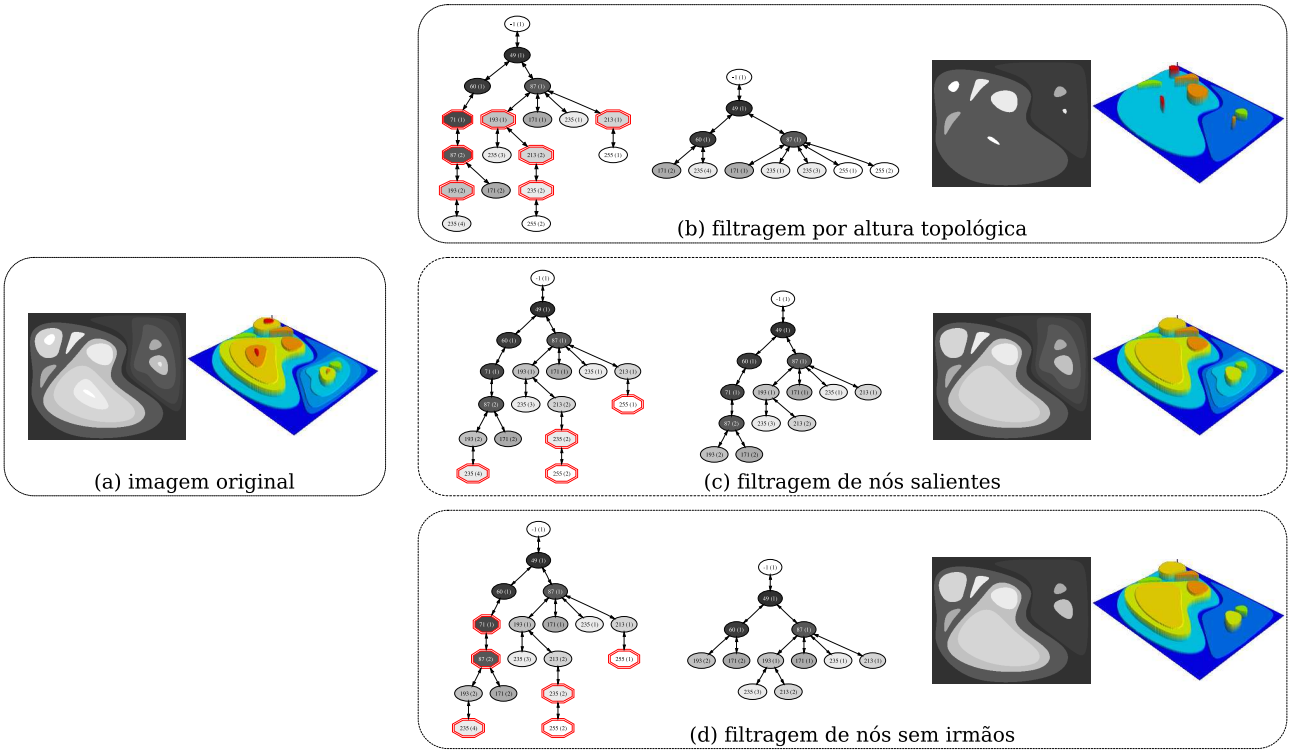


Fig. 4.6: Filtragem topológica. (a) Imagem original. (b) Remoção de nós com altura topológica (da subárvore) de 1 a 3. (c) Remoção de nós salientes. (d) Remoção de nós sem irmãos.

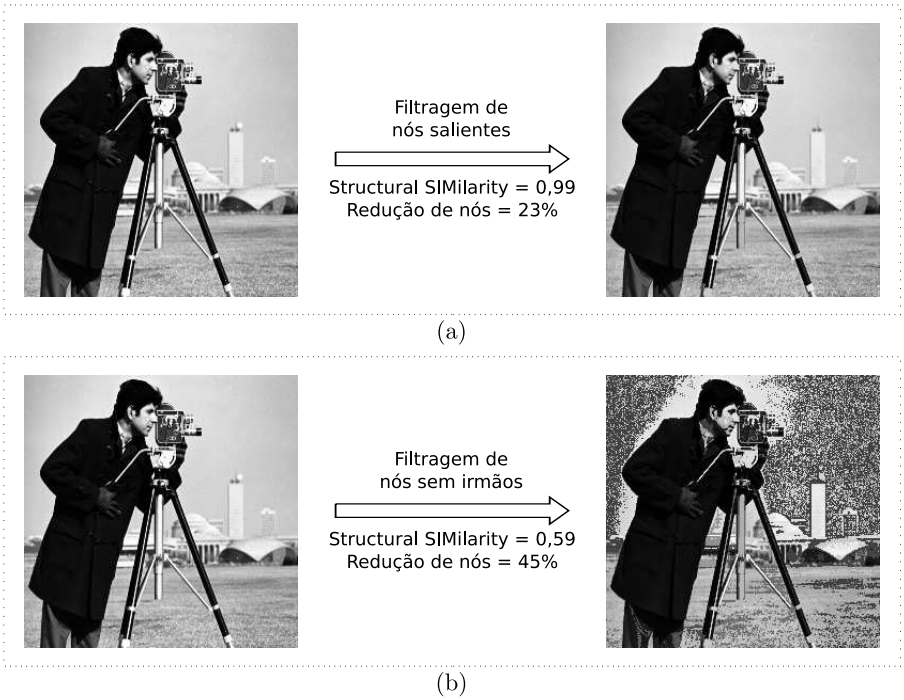


Fig. 4.7: Filtragem topológica. (a) Filtragem de nós salientes. (b) Filtragem de nós sem irmãos.

Restrição de contraste e limiarização baseada em regiões. A restrição de contraste é dada pela seleção de nós condicionada a sua escala local, $\mathcal{L}_C = \mathcal{N}_{n_{max}} - \mathcal{N}_{n_{min}}$, menor ou maior que uma constante pré-definida. Quanto à limiarização, o cálculo de Niblack [99] ou de Bernsen [100] pode ser definido, para cada componente, respectivamente, por $\mathcal{L}_N = \mathcal{N}_\mu + \alpha \cdot \mathcal{N}_\sigma$ (onde $\alpha \in [-1, +1] \in \mathbb{R}$ deve ser escolhido) e $\mathcal{L}_B = \frac{\mathcal{N}_{max} - \mathcal{N}_{min}}{2}$. Os nós são preservados ou não se tais valores estiverem acima ou abaixo de uma constante[3]. A Figura 4.8 mostra um exemplo sintético de ambos operadores. Na Figura 4.9, há um exemplo real, no qual, cada componente de nível da imagem de uma aspirina sofre restrição de contraste de 50. Verifica-se a simplificação da imagem de (a) para (c) e respectivos máximos regionais (b) e (d). Na Figura 4.10, por sua vez, observa-se a segmentação de biscoitos pela limiarização aplicada a componentes sugerida, em comparação com a limiarização de Otsu [100]. O limiar em função da estatística dos componentes, em (b) e binarização em (c), embora com escolha manual do valor 140 e $\alpha = -0,1$, produziu, em um único passo, a segmentação desejada com somente 2 componentes conexos referentes a regiões de interesse. Com o limiar de Otsu (d), foram obtidos 6 componentes conexos, dos quais, aqueles indesejados, neste caso, caracterizados como pequenas regiões localizadas próximas às bordas dos objetos, são facilmente removidos com uma abertura por área, mas com processamento extra, além da definição arbitrária da área a remover.

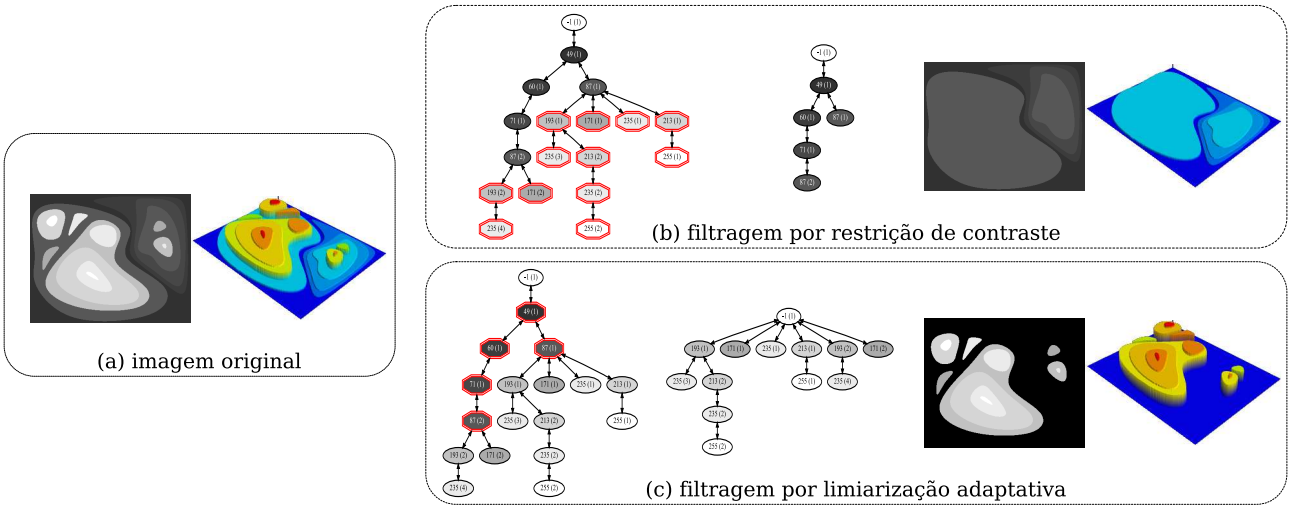
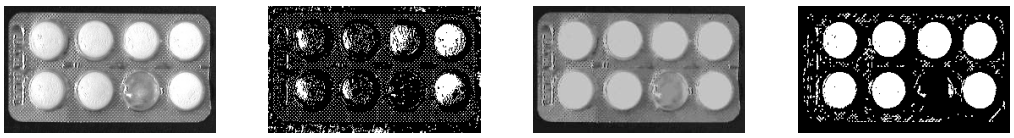


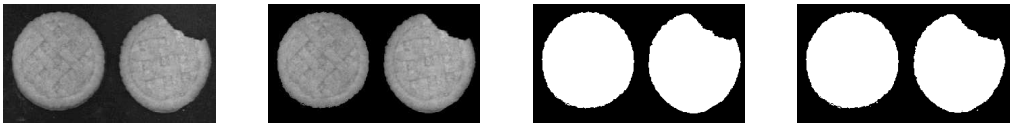
Fig. 4.8: Filtragem estatística. (a) Imagem original 322×402. (b) Remoção de componentes com contraste menor ou igual a 100. (c) Remoção de componentes com limiar 150 e $\alpha = -0,5$.

[3]Niblack e Bersen definem originalmente seus respectivos limiares em função da vizinhança de um pixel. Caso a intensidade deste esteja acima ou abaixo do limiar, assumirá valor 1 ou 0 na imagem resultante.



(a) Imagem original (b) Máximos de (a) (c) $\mathcal{L}_C \leq 50$ (d) Máximos de (c)

Fig. 4.9: Exemplo de restrição de contraste de componente.



(a) Original image (b) $\mathcal{L}_N < 140$, $\alpha = -0,1$ (c) $X_t(\mathcal{L}_N < 140)$, $t = 1$ (d) Otsu

Fig. 4.10: Exemplo de limiarização em componentes.

4.2 Operadores mistos

Filtros, com base nos atributos apresentados, podem misturar aspectos da topografia da imagem com a topologia da árvore. Nesta seção, três operadores são desenvolvidos com este perfil.

4.2.1 Áreas próximas

Na topografia da imagem, é comum ocorrer de componentes de nível, relacionados como pai $\mathcal{C}_{\mathcal{N}^P}$ e filho $\mathcal{C}_{\mathcal{N}^F}$, terem áreas (ou alturas ou volumes) próximas, ou seja, $\mathcal{N}^P_{area} \approx \mathcal{N}^F_{area}$, conforme ilustrado na Figura 4.11(a). Ao ser detectada a condição de $\mathcal{N}^P_{area} - \mathcal{N}^F_{area} \leq \Delta_{area}$, sendo Δ_{area} arbitrário, na análise da Max-tree, um dos dois nós, $\mathcal{C}_{\mathcal{N}^P}$ ou $\mathcal{C}_{\mathcal{N}^F}$, pode ser removido sem comprometer a imagem. A Figura 4.11(b) mostra um exemplo para $\Delta_{area} = 5000$ (valor este expressivo para haver algum efeito na imagem sintética composta por poucos componentes com áreas distantes entre si). A Figura 4.12 ilustra situações mais práticas deste filtro e demonstra, com dois exemplos, que, mesmo com redução significativa de nós (mais de 45%), a similaridade SSIM [97], entre as imagens antes e após esta filtragem, é alta (cerca de 99% para a imagem fotográfica). O Algoritmo 4.6 implementa a remoção de filho se tiver área próxima a do pai. O processo inicia-se pelas folhas mais "profundas" (linha 1), ou seja, na ordem descrescente de seus níveis topológicos (atributo definido no capítulo anterior), de modo que haja padronização, sem inconsistência, na sequência de eventuais remoções de componentes. As áreas de pai e filho, são observadas no caminho a partir de cada folha até a raiz (linhas $2-7$), havendo descarte de nó se a diferença destas for inferior ou igual a Δ_{area} (linha 4). O parâmetro $remove\_folha$ é um valor booleano que oferece a opção de remoção ou não de folhas (linha 5) que são, muitas vezes, elementos visuais notórios da imagem (na Fig. 4.11b, por exemplo, estas são preservadas).

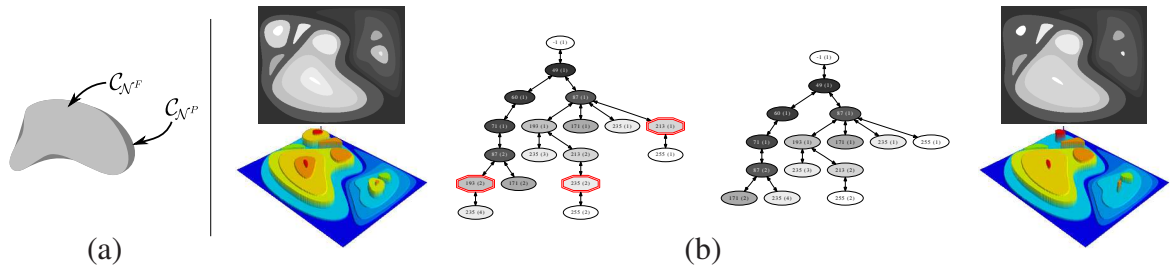


Fig. 4.11: Operador de áreas próximas entre pai e filho. (a) Ilustração de componentes "pai-filho" similares. (b) Exemplo de remoção de componentes para $\Delta_{area} = 5000$.

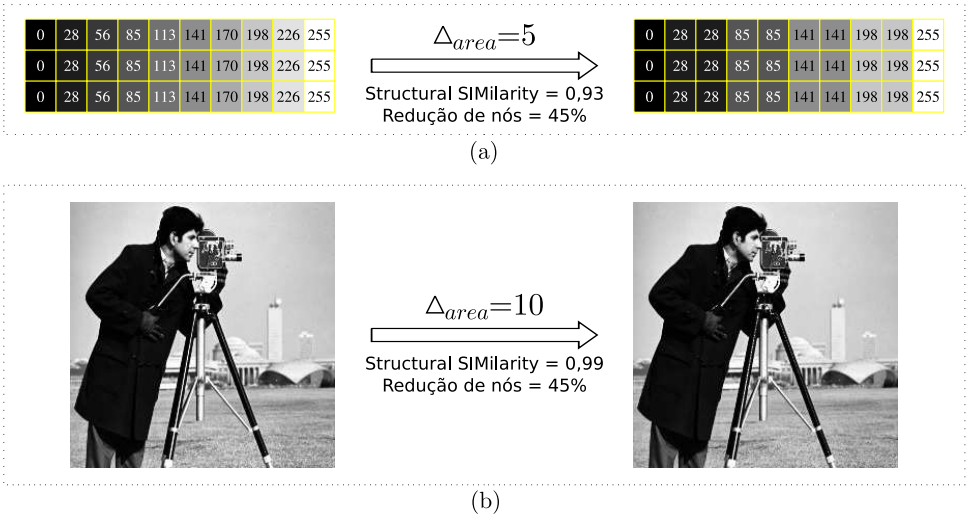


Fig. 4.12: Filtragem por áreas próximas entre pai e filho. (a) Imagem sintética de rampa. (b) Imagem fotográfica.

4.2.2 k-max

O objetivo principal deste operador [17] é o de obter regiões próximas a máximos regionais (ou mínimos regionais se a Min-tree for utilizada). A topologia da árvore é explorada ao se isolar nós com certa distância das folhas, e a topografia da imagem, ao se levar diferenças de níveis de cinza em consideração. O Algoritmo 4.7 detalha a ideia, na qual, a partir de cada folha (linha 1), segue-se k_{sobe} passos no caminho em direção à raiz (linha 7) até que o último nó visitado esteja a pelo menos k_{desce} passos da raiz da árvore (linha 3) – esta última restrição impede a eliminação de regiões normalmente maiores ou próximas à intensidade mínima da imagem –; no máximo, com H_{max} de diferença de nível de cinza em relação ao máximo regional (folha) de origem (linha 9); e enquanto de um filho para seu pai não houver um salto maior que h_{max} em relação à diferença entre seus níveis de cinza (linha 9). Na árvore da Figura 4.13(a), tem-se indicado a aplicação de tais parâmetros. Em (b), obtém-se todos os máximos regionais. Em (c), sobe-se 1 e mantém-se distância 2 da raiz. Em (d) e (e), limita-se o H_{max} e h_{max} respectivamente. Conforme a escolha dos parâmetros, o operador pode se comportar

Algoritmo 4.6: Algoritmo de simplificação da árvore com base na difererença de áreas entre pai e filho.

ENTRADA: MT_I, Δ_{area}, $remove\_folha$
SAÍDA: MT_I

ÁREAS_PRÓXIMAS()
1 **para cada** $\mathcal{N} \in$ (folhas de MT_I em ordem decrescente de nível topológico)
2 $\quad \mathcal{N}^P \leftarrow$ pai de $\mathcal{N}$
3 $\quad$ **enquanto** $\exists \mathcal{N}^{\mathcal{P}}$
4 $\quad\quad$ **se** $\mathcal{N}^P_{area} - \mathcal{N}_{area} \leq \Delta_{area}$
5 $\quad\quad\quad$ **se** $remove\_folha$ **e** $\mathcal{N}_{grau} = 0 \quad \Rightarrow \quad MT$.remove($\mathcal{N}$)
6 $\quad\quad$ **senão**
7 $\quad\quad\quad \mathcal{N} \leftarrow \mathcal{N}^P; \quad \mathcal{N}^P \leftarrow$ pai de $\mathcal{N}$
8 ATUALIZA_ATRIBUTOS(MT_I)

como: **máximo regional** se $k_{sobe} = 0$, $k_{desce} = 0$, $h_{max} = \infty$ e $H_{max} = \infty$; e H**-maxima** [38, 101] se k_{sobe} =(altura da árvore), $k_{desce} = 0$, $h_{max} = \infty$ e H_{max} qualquer. A Figura 4.14 exemplifica uma aplicação de segmentação da região porosa em imagens de solo, obtidas por um tomógrafo [102], com base no k-max [17].

Algoritmo 4.7: Algoritmo k-max.

ENTRADA: MT_I, k_{sobe}, k_{desce}, H_{max}, h_{max}, $particao$, $finais$
SAÍDA: MT_I

K-MAX()
1 **para cada** $\mathcal{N} \in$ (folhas de MT_I)
2 $\quad n_{inicial} \leftarrow \mathcal{N}_{nivel}$
3 $\quad$ **se** $k_{desce} \leq \mathcal{N}_{ntop}$
4 $\quad\quad$ **se** (**não** $finais$) $\quad \Rightarrow \quad \mathcal{N}_{particao} = particao$
5 $\quad\quad$ **se** $k_{sobe} < \mathcal{N}_{ntop} - k_{desce} \quad \Rightarrow \quad passos \leftarrow k_{sobe}$
6 $\quad\quad$ **senão** $\quad \Rightarrow \quad passos \leftarrow \mathcal{N}_{ntop} - k_{desce}$
7 $\quad\quad$ **para** $k = 0$ **até** $passos - 1$
8 $\quad\quad\quad n_{anterior} \leftarrow \mathcal{N}_{nivel}; \quad \mathcal{N} \leftarrow$ pai de $\mathcal{N}$
9 $\quad\quad\quad$ **se** $(n_{anterior} - \mathcal{N}_{nivel} > H_{max})$ **ou** $(n_{anterior} - \mathcal{N}_{nivel} > h_{max}) \quad \Rightarrow$ **quebre laço**
10 $\quad\quad\quad$ **senão, se** (**não** $finais$) $\quad \Rightarrow \quad \mathcal{N}_{particao} \leftarrow particao$
11 $\quad\quad$ **se** $(k_{sobe} = 0$ **ou** $k > 0)$ **e** $finais \quad \Rightarrow \quad \mathcal{N}_{particao} \leftarrow particao$
12 ATUALIZA_ATRIBUTOS(MT_I)

4.2.3 Reconstrução morfológica

A reconstrução morfológica de marcadores S, condicionados a uma imagem I, é um tema recorrente em processamento de imagens [45, 103, 85], aplicada normalmente como ferramenta auxiliar

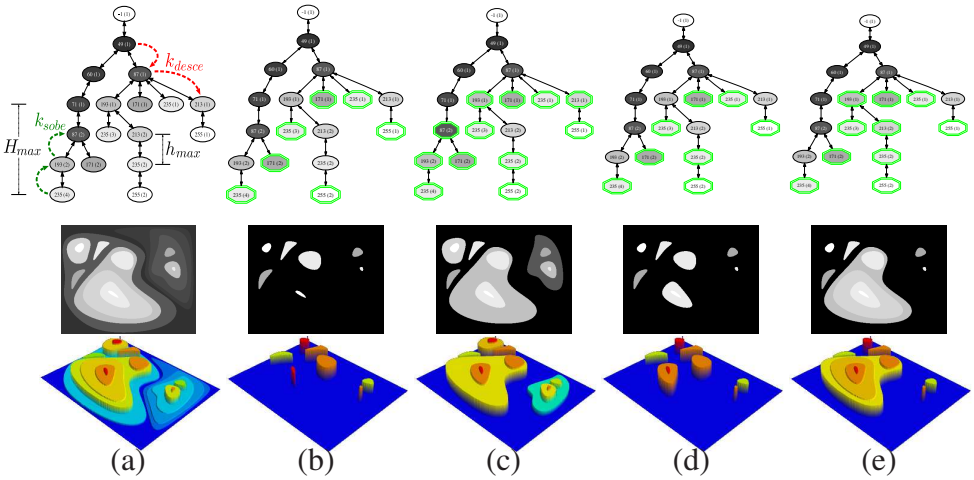


Fig. 4.13: Operador k-max. (a) I e parâmetros. (b) Seleção $k_{sobe} = 0$ e $k_{desce} = 0$. (c) Seleção $k_{sobe} = 1$ e $k_{desce} = 2$. (d) Seleção $k_{sobe} = 5$, $k_{desce} = 0$ e $H_{max} = 40$. (e) Seleção $k_{sobe} = 5$, $k_{desce} = 0$ e $h_{max} = 40$.

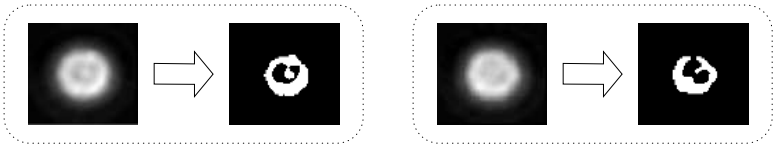


Fig. 4.14: Exemplo de aplicação do k-max com $k_{sobe} = 5$ e $k_{desce} = 5$.

para realçar elementos específicos (marcados), atenuando o restante da imagem. Pode-se pensar neste operador como uma sequência infinita de dilatações de S (até a estabilidade, na prática) condicionadas a I [38]. No entanto, esta ação apresentaria custo computacional elevado dado o encadeamento de filtros não-lineares. Uma forma mais eficiente é interpretada como a manutenção, na imagem I, apenas de componentes que têm intersecção com o marcador S (vide Eq. 2.11). Em vez de gerar uma imagem resultante da reconstrução, o Algoritmo 4.8 implementa tal ideia no contexto da Max-tree. A Figura 4.15 ilustra dois exemplos de reconstrução, o primeiro com um único pixel marcador (a) e o segundo com dois marcadores (b). A reconstrução morfológica, nestes moldes, requer marcadores na imagem e processamento na árvore e trata-se, portanto, de um operador com elementos de ambos domínios. A proposta deste trabalho consiste em uma implementação alternativa sob ponto de vista da Max-tree.

O desempenho do algoritmo de reconstrução, baseado na Max-tree, é satisfatório, mesmo havendo necessidade de criação da árvore, processamento para preservação dos nós de interesse e reconstrução (renderização) da imagem de saída. A Tabela 4.1 (no final do capítulo) exibe uma comparação entre operador idêntico com implementação usando varreduras *raster*, *anti-raster* e fila [45], em C++, pela

Algoritmo 4.8: Algoritmo de reconstrução, a partir de um marcador S, feita no domínio da árvore.

RECONSTRUÇÃO_MORFOLÓGICA(MT_I, S)
1 **para cada** $p \in S$
2 $\quad \mathcal{N} \leftarrow \mathcal{N}_{I[p]}^{status[p]}$
3 $\quad$ **enquanto** $\exists \mathcal{N} \quad \Rightarrow \quad$ (marca $\mathcal{N}$ como pertencente à reconstrução; $\quad \mathcal{N} \leftarrow$ pai de $\mathcal{N}$)
4 **para cada** $\mathcal{N} \in MT_I$
5 $\quad$ **se** $\mathcal{N}$ não está marcado como visitado $\quad \Rightarrow \quad MT_I$.remove$(\mathcal{N})$
6 ATUALIZA_ATRIBUTOS(MT_I)

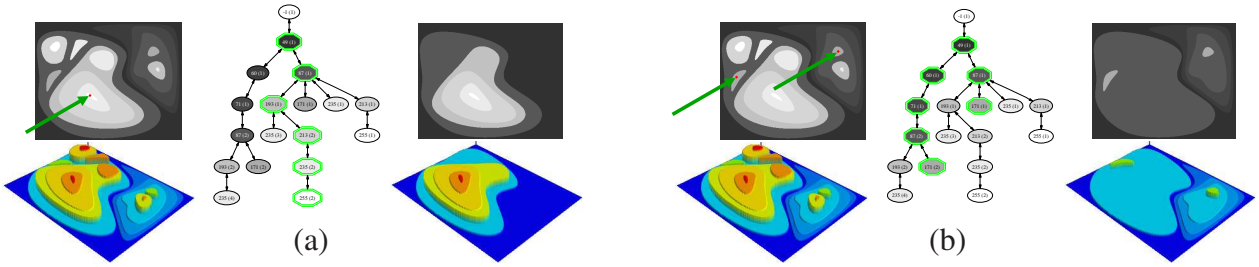


Fig. 4.15: Operador de reconstrução. (a) A partir do pixel $(200, 150)$ ou nó $\mathcal{C}^2_{255}$. (b) A partir dos pixels $(150, 45)$, $(80, 325)$ ou nós $\mathcal{C}^1_{171}, \mathcal{C}^2_{171}$.

eficiente biblioteca SDC Morphology Toolbox[4], na qual não há utilização da Max-tree; e a proposta aqui apresentada. As colunas, da esquerda para a direita, são: imagem original, disposição dos marcadores – primeiro teste com apenas um ponto central; segundo teste com toda a linha central; e terceiro teste com um *frame* de pontos, ou seja, primeira e última linhas e primeira e última colunas –, nós remanescentes da reconstrução, tempo para a solução do estado da arte sem uso da Max-tree, primeira proposta^a – com a árvore sem qualquer alteração, contendo a totalidade dos atributos definidos neste texto – e seu ganho e, por fim, a segunda proposta^b – com árvore modificada, contendo recursos suficientes para implementação da reconstrução – e seu ganho sobre a solução da SDC.

Observa-se que, para uma mesma imagem, o acréscimo de pontos nos marcadores tende a reduzir o tempo de processamento na solução sem a Max-tree, dada a preservação de uma quantidade maior de pixels da imagem e maior brevidade na estabilização do processo de crescimento de regiões. Enquanto o desempenho das propostas, com a Max-tree, se mantém com alteração mínima, pois tanto o algoritmo de construção da árvore (Alg. 3.1) ou de renderização (Alg. 4.3), no final do processo, visitam, em tempos praticamente iguais (em segundos), todos os pixels desta mesma imagem, representando a maior fatia de processamento (a construção da Max-tree, por exemplo, significa cerca de três quartos do tempo total de processamento desta forma de reconstrução morfológica). De todo modo, a grande vantagem do método sugerido está na disponibilidade da árvore simplificada, resultante da preservação apenas de nós relacionados à reconstrução, para eventuais processamentos

[4] `http://mmorph.com`

subsequentes antes da geração da imagem de saída.

4.3 Complexidade

Verificou-se, no Capítulo 3, a eficiência de dois destacados algoritmos para construção da Max-tree. Na prática, ambos se caracterizam por tempo quase-linear supondo um valor relativamente baixo para o fator multiplicatvo de suas complexidades ($\alpha(n) < 4$ para o algoritmo de [36] ou $2 \leq L \leq 256$ para o algoritmo de [1]). Tem-se claro também que, no domínio da árvore, há menos elementos a processar (por exemplo, para 49 imagens 512×512 da base `http://decsai.ugr.es/cvg/CG/base.htm`, o número de nós corresponde, em média, a 16% do número de pixels) e o desempenho passa a ser medido em função do número de nós N. Supondo que cada nó possa ter, no máximo m filhos, e que a altura topológica seja h, então uma árvore completa apresenta m^h folhas. Ainda para uma árvore completa, o número de nós em função da altura consiste em uma soma de progressão geométrica (dado o nível 0 com 1 nó, nível 1 com m nós, ..., nível k com m^k nós) igual a $N = \frac{m^{h+1}-1}{m-1}$. Isolando h nesta expressão, obtém-se $h = \log_m[N(m-1)+1] - 1$.

A atualização de atributos (Alg. 4.2), utilizado em todas as operações de filtragem de nós, percorre, a partir de cada folha, todos os nós ancestrais até a raiz, sendo, portanto necessárias $m^h h$ operações. Para uma árvore completa, esse valor é $m^{\{\log_m[N(m-1)+1]-1\}}\ \{\log_m[N(m-1)+1]-1\}$, ou seja, $\frac{N(m-1)+1}{m}\ \{\log_m[N(m-1)+1]-1\}$ operações. Sendo assim, para valores assintóticos de N e m, é fácil demonstrar que o referido algoritmo é $O(N \log N)$ (caso médio). Supondo a configuração de nós dada pela Figura 4.16, na qual a árvore tem $N/2$ folhas e altura $N/2$, tem-se então $(N/2)^2$ operações, ou seja, $O(N^2)$ (pior caso). Porém, na prática, a altura máxima da árvore é $L-1$, pois L (quantidade de níveis de cinza) limita o tamanho do "empilhamento" de componentes de nível na Max-tree. Desta forma, a complexidade é, no pior caso, $O(N \times L)$, sendo L tipicamente constante.

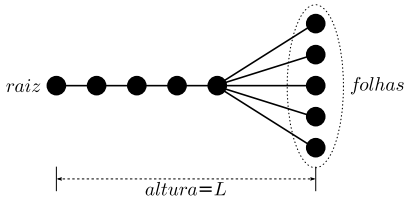


Fig. 4.16: Topologia da Max-tree, em "pior caso", para atualização de atributos.

Quanto à filtragem de atributos (Alg. 4.1), esta efetua uma visitação de todos os nós, em tempo linear, seguida pela atualização dos atributos. Desta forma, também se caracteriza como $O(N \times L)$. A filtragem de nós salientes (Alg. 4.4) verifica, a partir de cada folha, o primeiro ancestral com mais de um filho, sendo, no pior caso, $O(N \times L)$ e, em seguida, também procede a atualização de atributos. Já a filtragem de nós sem irmãos (Alg. 4.5) é feita em tempo linear, porém o algoritmo passa a $O(N \times L)$

devido à atualização de atributos. Na simplificação de nós, cujas áreas entre pai e filho são próximas (Alg. 4.6), é necessária a ordenação das folhas pelo atributo "nível topológico" em $O(N \log N)$ e, a partir de cada folha, verificação das relações "pai-filho" dos ancestrais que, no pior caso, é feita em $O(N \times L)$. Desta forma, este algoritmo é de ordem $O(N \log N)$, supondo L constante e $O(N \times L)$ caso contrário. O k-max (Alg. 4.7), por sua vez, caminha em direção à raiz, no pior caso, k passos a partir de cada folha e, portanto, o algoritmo é $O(k \times N)$, sendo $0 \leq k < L$, além da atualização de atributos.

Duas operações necessitam de informações no domínio da imagem, onde seu número n de pixels deve ser levado em consideração. Na reconstrução da imagem (Alg. 4.3), uma varredura (*raster*) é feita na imagem e, para cada pixel visitado, verifica-se em tempo constante (pelo *hashing*) o nó associado na árvore. Caso este tenha sido filtrado, procura-se o primeiro ancestral ativo. Este procedimento leva, no pior caso, $O(n \times L)$. Por fim, a reconstrução morfológica (Alg. 4.8), consiste em determinar os nós a partir das sementes, em pior caso, para todos os n pixels da imagem, ou seja, em tempo linear. E, para cada um destes nós, visita-se todos os $L - 1$ ancestrais preservando-os no resultado. Varre-se posteriormente todos os nós, em tempo linear, removendo aqueles sem indicação de preservação. Ao final, tem-se complexidade $O(n \times L)$ para a reconstrução morfológia.

Para todos os operadores mencionados, a atualização de atributos pode eventualmente não ser necessária, caso a intenção seja apenas a de reconstrução da imagem sem os componentes eliminados, após a última filtragem (ou seja, não havendo filtragens subsequentes). A Tabela 4.2 resume o desempenho dos algoritmos com ou sem esta tarefa de atualização.

4.4 Considerações sobre o capítulo

Neste capítulo, um conjunto de filtros é apresentado e exemplos diferenciados de resultados produzidos são exibidos. A peculiaridade é que todos eles são baseados na Max-tree. Podas e enxertos são efetuados a partir de propriedades dos nós. Alguns filtros novos são estabelecidos considerando, por exemplo, a verificação de atributos topológicos, acrescentados em tempo de construção da árvore, como nível topológico, número de filhos (grau) ou número de descendentes de cada nó. A ideia da remoção de nós salientes e de nós sem irmãos (este último contemplando o primeiro), por exemplo, é a de simplificar a árvore (imagem) para processamentos seguintes [30]. A redução de nós da Max-tree, usando o filtro "nós salientes", não é significativa , porém há manutenção do aspecto visual da imagem (como visto na Fig. 4.7a). Em relação aos "nós sem irmãos", uma quantidade mais ampla de nós é retirada e as ramificações são preservadas, porém com comprometimento da similaridade entre imagens de pré e pós-processamento (Fig. 4.7b). A remoção de nós por "áreas próximas" é desenhada no sentido de possibilitar ambas características desejáveis: a simplificação efetiva da árvore

e menor alteração possível de percepção visual da imagem renderizada (Fig. 4.12). A simplificação também ocorre nos filtros estatísticos sugeridos, baseada na restrição de nós segundo a composição de intensidades em cada componente de nível. Utilizando atributos como mínimo, máximo, média e desvio padrão de níveis de cinza, produzidos incrementalmente no algoritmo de inundação do [1], pode-se estabelecer restrições de contraste ou limiarizações a cada componente da imagem.

A supressão de nós impacta no tempo de processamento de etapas subsequentes na mesma árvore (pois há menos elementos a considerar) com resultado semelhante ao obtido sobre a árvore original (com a totalidade dos elementos). Além disso, verifica-se que as imagens geradas após filtragens, como, por exemplo, a de "áreas próximas", apresentam compressão sempre maior que a original, seja aplicando um algoritmo genérico (como os baseados em codificação de Huffman) ou outros específicos para imagens (como os baseados em transformada discreta de cossenos), devido à presença de menor número de componentes de nível e, consequentemente, da elevação de ocorrências das intensidades que persistiram na imagem resultante. O operador k-max [17], por sua vez, permite uma flexibilização de escolha de parâmetros, em relação à quantidade de arestas a partir das folhas da árvore ou às diferenças entre níveis de cinza nas relações topológicas, para seleção de regiões ao redor de máximos regionais. Este recurso pode se especializar, por exemplo, no operador H-maxima [38]. Quanto à reconstrução morfológica, uma grande vantagem do método, baseado na Max-tree, está na disponibilização de uma estrutura simplificada, resultante da preservação apenas de nós pertinentes, para eventuais processamentos subsequentes antes da geração da imagem de saída. O tempo de processamento desta forma de reconstrução morfológica pode eventualmente ser vantajoso se o número de pontos de marcação for suficientemente pequeno (vide Tab. 4.1). Métodos clássicos de crescimento de regiões, nestes casos, acabam tendo um número de iterações elevado até a estabilidade.

Tab. 4.1: Comparação de implementações de reconstrução sem e com utilização da Max-tree.

Entrada	Marcadores	Saída	Nós da rec.	Sem Max-tree	Com Max-tree[a]	Ganho de tempo[a]	Com Max-tree[b]	Ganho de tempo[b]
campinas	*ponto central*		0,43%	$0,09s$	$0,09s$	0%	$0,08s$	11%
	*linha central*		2,73%	$0,08s$	$0,11s$	-27%	$0,08s$	0%
500×324	*frame*		5,34%	$0,03s$	$0,10s$	-70%	$0,08s$	-63%
apucarana	*ponto central*		0,16%	$0,22s$	$0,23s$	-4%	$0,16s$	27%
	*linha central*		1,02%	$0,16s$	$0,23s$	-30%	$0,15s$	6%
800×600	*frame*		2,93%	$0,08s$	$0,24s$	-67%	$0,16s$	-50%
sanfrancisco	*ponto central*		0,07%	$0,31s$	$0,53s$	-42%	$0,38s$	-18%
	*linha central*		0,50%	$0,29s$	$0,53s$	-45%	$0,37s$	-22%
1280×980	*frame*		3,06%	$0,22s$	$0,54s$	-59%	$0,38s$	-42%
joinville	*ponto central*		0,00%	$0,12s$	$0,79s$	-85%	$0,60s$	-80%
	*linha central*		0,27%	$0,59s$	$0,79s$	-25%	$0,58s$	2%
1600×1200	*frame*		1,33%	$0,39s$	$0,80s$	-51%	$0,60s$	-35%
rio	*ponto central*		0,03%	$2,77s$	$1,42s$	49%	$1,08s$	61%
	*linha central*		0,42%	$2,78s$	$1,43s$	49%	$1,06s$	62%
2048×1536	*frame*		1,43%	$1,54s$	$1,43s$	7%	$1,07s$	31%
saoludgero	*ponto central*		0,01%	$3,82s$	$2,98s$	22%	$2,22s$	42%
	*linha central*		0,21%	$3,33s$	$2,96s$	11%	$2,16s$	35%
3008×2000	*frame*		1,11%	$1,82s$	$2,96s$	-39%	$2,20s$	-17%
Total				$18,64s$	$18,16s$	3%	$13,41s$	28%

Tab. 4.2: Complexidade dos operadores desenvolvidos com base na Max-tree, sendo n o número de pixels, N o número de nós, L a quantidade de níveis de cinza e $k \in \mathbb{N}^*$, $k < L$.

Algoritmo	Com atualização de atributos	Sem atualização de atributos
ATUALIZA_ATRIBUTOS	$O(N \times L)$	–
FILTRA_MAX-TREE_ATRIBUTOS	$O(N \times L)$	$O(N)$
FILTRA_SEM_IRMÃOS	$O(N \times L)$	$O(N)$
FILTRA_SALIENTES	$O(N \times L)$	$O(N \times L)$
ÁREAS_PRÓXIMAS	$O(N \log N)$	$O(N \log N)$
K-MAX	$O(N \times L)$	$O(k \times L)$
RECONSTRUÇÃO_IMAGEM	$O(n \times L)$	$O(n \times L)$
RECONSTRUÇÃO_MORFOLÓGICA	$O(n \times L)$	$O(n \times L)$

Capítulo 5

Valores de extinção

A localização e segmentação de objetos de interesse de forma mais direta possível é uma característica desejável em muitas aplicações de visão computacional. Em morfologia matemática, a definição de contornos de objetos é frequentemente obtida por *watershed* – ou algoritmo de linhas divisoras de água – [104] baseado na inundação a partir de marcadores. Considerando uma imagem em níveis de cinza como uma superfície topográfica (intensidade como valor de altitude), os platôs (zonas planas) na base de vales (mínimos regionais) ou no topo de montanhas (máximos regionais) são tipicamente utilizados como marcadores. Entretanto, imagens adquiridas por câmeras ou *scanners* normalmente apresentam um número acentuado destes pontos extremos, que podem se caracterizar como ruído ou outra região indesejável e gerar uma sobressegmentação. A seleção de mínimos ou máximos mais significativos como marcadores pode ser feita por dinâmica [105, 2, 8], uma medida de contraste dada pela mínima altitude a ser superada, no caminho, a partir de um vale, para atingir outro vale mais profundo, como ilustrado na Figura 5.1. Este conceito foi estendido pela definição de valores de extinção associados a cada extremo da imagem [106, 107]. O valor de extinção de um extremo regional (mínimo ou máximo), para qualquer atributo crescente (altura, área, volume, etc) é o tamanho máximo do filtro por atributo [10] tal que este extremo ainda permaneça após a filtragem [106]. Dinâmica, portanto, é um caso particular de extinção para o atributo "altura" (ou "profundidade"). Neste trabalho, o valor de extinção de máximos regionais é utilizado, de forma a aproveitar a configuração dos nós da árvore de componentes, e a interpretação, por sua vez, refere-se a um filtro por atributo suficientemente grande (remoção de considerável camada de terra de cumes) para o desaparecimento de uma montanha do relevo da imagem. Neste capítulo, são propostos cinco novos valores de extinção [9, 108]; dois baseados na topologia da árvore de componentes: **(i)** extinção de número de descendentes, **(ii)** extinção de altura topológica; e três geométricos: **(iii)** extinção de altura, **(iv)** de largura e **(v)** de diagonal da caixa mínima envolvente (*bounding box*). Estas extinções são eficientemente determinadas a partir do cálculo incremental de novos atributos para cada nó da árvore

– número de descendentes, altura de subárvore, coordenadas do canto superior esquerdo e canto inferior direito da caixa envolvente para o componente de nível associado – no processo de construção da Max-tree, mantendo o desempenho assintótico de soluções recentes da literatura[1]. Estas medidas permitem a seleção de marcadores relevantes, conforme a aplicação, diversificando as possibilidades de filtragens e segmentações. A principal contribuição deste trabalho é o estabelecimento de novos valores de extinção – além de altura [105], área e volume [106, 107] –, que possam ser usados como técnica única ou ferramenta auxiliar na identificação de objetos de interesse. A eficiência do algoritmo genérico para determinação de um valor extinção, assim como sua utilidade em imagens reais, são demonstradas neste capítulo.

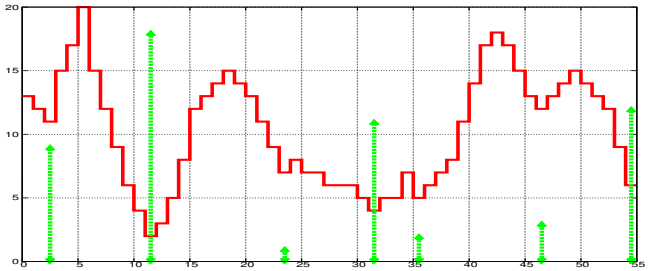


Fig. 5.1: Dinâmica de mínimo regional (linhas tracejadas) de uma imagem (linha contínua).

5.1 Dinâmica de máximos

Definição 5.1 (Dinâmica de caminho) Dinâmica de caminho*, supondo P_{x_a,x_b} que liga os* pixels x_a *e* x_b *de uma imagem* I*, é a diferença de níveis entre os pontos de mais alta e baixa intensidade neste caminho, ou seja:*

$$D_C(x_a, x_b) = \max\{|I(x_i) - I(x_j)| \mid x_i, x_j \in P_{x_a,x_b}\} \tag{5.1}$$

Definição 5.2 (Dinâmica entre pixels) Dinâmica entre dois pixels*, x_a e x_b, é igual à menor dinâmica de caminho entre eles, ou:*

$$D_P(x_a, x_b) = \min\{D_C(x_a, x_b) \mid P_{x_a,x_b} \textit{ é um dos possíveis caminhos entre } x_a \textit{ e } x_b\} \tag{5.2}$$

Definição 5.3 (Dinâmica de máximo regional) Dinâmica de um máximo regional[2] M *é a mínima altitude (diferença de níveis de cinza) que se deva descer, dado um caminho P_{x_a,x_b}, a partir de um*

[1] A árvore de componentes pode ser construída em tempo quase-linear [36].

[2] [105] apresenta dinâmica de mínimos regionais com lógica inversa a esta definição. O uso dos máximos regionais está de acordo com as características da Max-tree. Problemas podem também ser modelados na Max-tree do negativo da imagem ou Min-tree diretamente.

pixel x_a *de* M*, para se atingir um* pixel x_b *de um outro máximo regional* M_V *mais alto que* M*, ou seja:*

$$D_M = \min\{D_P(x_a, x_b) \mid x_a \in M, x_b \in M_V, I(x_a) < I(x_b)\} \tag{5.3}$$

A Figura 5.2 ilustra esta ideia. M_i, $i \in [1, 6]$, são os máximos regionais. Para a determinação da dinâmica[3] de M_2, há dois máximos mais altos, M_1 e M_5. No entanto, é preciso descer h_1 no caminho do primeiro e h_2, no segundo. Sendo $h_2 < h_1$, o segundo caminho (tracejado) é preferido, e $D_{M_2} = h_2$. A dinâmica é uma medida de contraste concisa e poderosa para identificação de regiões de interesse na imagem. Exemplos são exibidos nas seções seguintes.

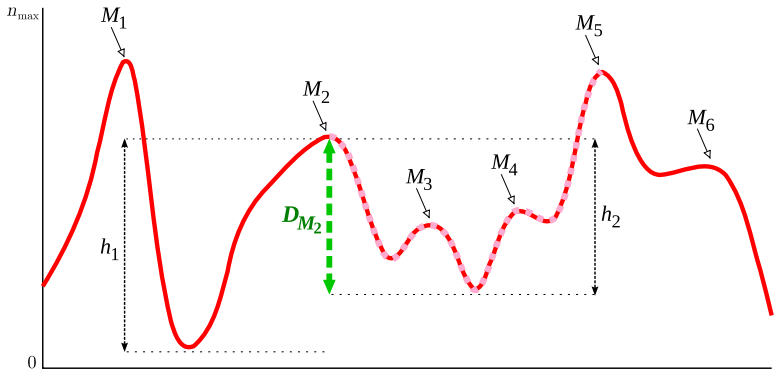


Fig. 5.2: Dinâmica do máximo regional M_2.

5.2 Valores de extinção

Na seção anterior, a dinâmica foi discutida. Em suma, esta mede a menor redução de altitude, a partir de uma máximo regional, para se alcançar outro máximo regional mais alto. Em outras palavras, trata-se da *extinção de altura* de uma elevação ou montanha no relevo (ou subárvore na Max-tree) para λ_h suficientemente grande em um filtro por atributo $\Upsilon_{altura,\lambda_h}(I)$ (poda da Max-tree). Este conceito pode ser estendido para outros atributos, além de altura (relacionada à diferença de níveis de cinza), desde que os mesmos sejam crescentes no sentido dos máximos regionais (folhas) até o menor nível de cinza (raiz) presente na imagem (árvore). Acima de um componente de nível (nó $\mathcal{N}$ da árvore) há um "bloco de terras" do relevo da imagem (nós descendentes de $\mathcal{N}$) em que se pode determinar, além da altura, a área e volume (vide 2.2.1). Todos estes atributos aumentam se for considerado um componente de nível obtido pela limiarização em um nível de cinza menor (por exemplo, o nó pai de $\mathcal{N}$ apresenta altura, área e volume maiores). Desta forma, assim como

[3]Na ausência de especificação, trata-se de *dinâmica de máximos regionais*.

extinção de altura, pode-se definir *extinção de área* – partindo de um máximo, a área final da base da elevação para se alcançar outra elevação (máximo regional) cuja base tenha área maior, ou $\Upsilon_{area,\lambda_a}(I)$ com λ_a suficientemente grande para o desaparecimento da primeira elevação –, e *extinção de volume* – partindo de um máximo, o volume final da elevação para se alcançar outra elevação (máximo regional) com volume maior, ou $\Upsilon_{volume,\lambda_v}(I)$ com λ_v suficientemente grande para o desaparecimento da primeira elevação. Segue definição formal generalizada.

Definição 5.4 (Extinção de máximo regional) *A* extinção de um máximo regional M*, em relação a um atributo crescente* μ*, é o menor valor possível de* λ *para que, após o filtro por atributo* $\Upsilon_{\mu,\lambda}(I)$ *(definida pela Eq. 2.12), não haja mais um máximo regional (Eq. 2.7), no qual* M *esteja contido:*

$$E_\mu(M) = \min\{\lambda \geq 0 \mid M \cap Maxreg\left(\Upsilon_{\mu,\lambda}(I)\right) = \emptyset\} \tag{5.4}$$

[106] estabelecem valores de extinção para área e volume de mínimos regionais. Neste trabalho, máximos regionais são usados, por serem facilmente extraídos a partir da construção eficiente da Max-tree, e novas extinções são propostas.

5.3 Extinções infinitas ou empatadas

Duas particularidades podem ocorrer na determinação de um valor de extinção baseada em um atributo μ qualquer: **(i) infinito** quando, em relação ao máximo regional M_a tratado com atributo crescente μ_a, não houver outro máximo M_b com atributo μ_b maior, ou seja, $\forall M_b \neq M_a, \nexists \mu_b > \mu_a$ e, neste caso, o relevo da imagem se extingue como um todo. Na prática, ocorre uma filtragem por atributo $\Upsilon_{\mu,\lambda_{\max}}(I)$, no qual $\lambda_{\max}$ é o máximo valor assumido pelo atributo μ considerado; **(ii) empate** se dois ou mais máximos regionais distintos, M_a tratado e M_b qualquer, posicionados sobre uma mesma "montanha", têm mesma extinção[4]. Neste caso, convém considerar diferenças infinitesimais entre atributos, ou seja, $\mu_a > \mu_b$ e $\mu_a - \mu_b \approx 0$, sendo, portanto, atribuída a extinção original apenas a M_a. Qualquer M_b, inicialmente empatado, assume extinção menor que M_a com atributo infinitesimalmente maior. Com isto, pretende-se ter um aumento da região de influência dos máximos mais representativos, reduzindo a ocorrência de possíveis marcadores para a localização de objetos. A Figura 5.3 exemplifica, por meio da dinâmica (extinção de altura) de máximos regionais, a resolução de extinção infinita para M_1, máximo mais alto (maior nível de cinza), cujo valor de extinção de altura 20 faz desaparecer todas as elevações do relevo; e o empate inicial entre as extinções dos máximos M_2, M_3 e M_4 por terem mesma altitude (nível de cinza). Todos, a princípio, teriam extinção 15 em relação a M_1, mas a resolução de empate considera M_4 infinitesimalmente mais alto que M_3

[4] Análogo aos mínimos regionais com altitudes idênticas, sob um mesmo vale, do trabalho de [105].

e este, por sua vez, minimamente superior a M_2 e, portanto, M_2 assume extinção 5 em relação a M_3 e, este, 10 em relação a M_4. Deste modo, M_4, agora o único com extinção 15 (em relação a M_1) exerce influência sobre a elevação do relevo, na qual também estão incluídos M_2 e M_3, e seria considerado, em termos de dinâmica, como o segundo mais significativo desta imagem. Neste trabalho, dentre os máximos empatados sob dada extinção, a eleição daquele que deverá possuir extinção infinitesimalmente maior, tem relação com a ordem de visitação de pixels do algoritmo de construção da Max-tree utilizado. A definição arbitrária de ordem entre os máximos, em vez desta diferenciação baseada em níveis de cinza e sequência de visitação de pixels, é uma outra forma de evitar empates [8], porém, a parametrização seria complexa e os resultados práticos eventualmente desvantajosos, se mais de um máximo assumir extinção relevante para uma mesma região, condição esta não almejada se a intenção é a de simplificação de máximos significativos.

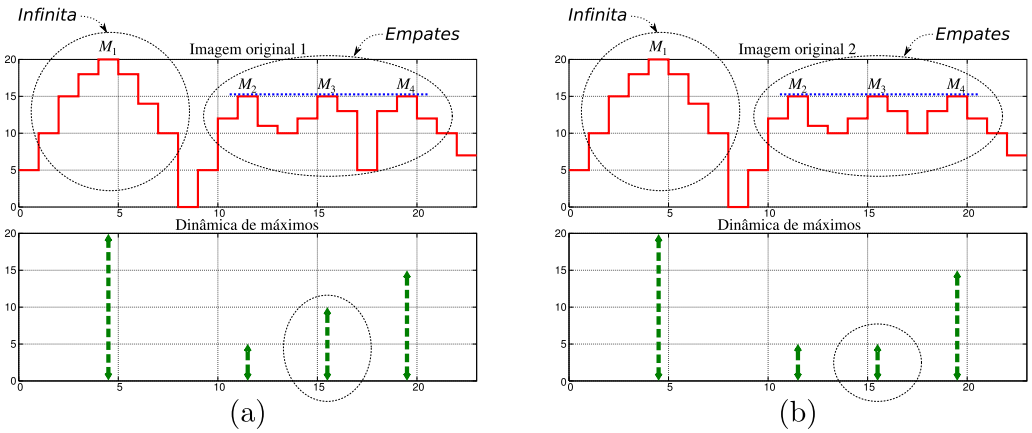


Fig. 5.3: Extinções de altura – infinita e empatadas – de máximos regionais. (a) Definição de ordem dos atributos (alturas) $\mu(M_2) < \mu(M_3) < \mu(M_4)$. (b) Mesma diferenciação infinitesimal de alturas nos empates, e mesmo valor para os mínimos entre M_2 e M_3, e entre M_3 e M_4.

Seguem definições matemáticas da terminologia discutida nesta seção.

Definição 5.5 (Extinção infinita) *A* extinção infinita E_μ^∞*, em relação a um atributo crescente* μ*, está associada ao máximo regional com maior atributo* μ*, e consiste no valor mínimo de* λ *para que, após a filtragem por atributo* $\Upsilon_{\mu,\lambda}(I)$*, não haja mais nenhum máximo regional na imagem (uma única zona plana permanece):*

$$E_\mu^\infty(I) = \min\{\lambda \geq 0 \mid \mathit{Maxreg}\,(\Upsilon_{\mu,\lambda}(I)) = \emptyset\} \tag{5.5}$$

Definição 5.6 (Empate de extinções) *O* empate de extinções *acorre quando dois ou mais máximos regionais, com igual atributo* μ*, apresentam mesmo valor de extinção* E_μ *e permanecem contidos em um mesmo máximo regional* r *de* $\Upsilon_{\mu,\lambda}(I)$ *(internos a uma mesma elevação do relevo da imagem),*

até que λ seja suficiente para suas extinções simultaneamente. Portanto, dois máximos regionais distintos, M_a e M_b, têm empate de extinções, se λ do filtro por atributo tiver sido escolhido sob a condição:

$$E_{\mu}^{=}(M_a, M_b) = \min\{\lambda \geq 0 \mid \mu(M_a) = \mu(M_b),\ M^* \cap M_a = \emptyset,\ M^* \cap M_b = \emptyset\} \tag{5.6}$$

sendo, $M^* = \mathcal{C}_1^r\left(Maxreg\left(\Upsilon_{\mu,\lambda}(I)\right)\right)$.

Definição 5.7 (Desempate de extinções) *O* desempate de extinções *consiste no estabelecimento de uma ordenação de máximos regionais M_i empatados entre si, em termos de valor de extinção, dada pela diferenciação infinitesimal de seus atributos:*

$$\mu(M_i) = \mu(M_{i+1}) + \epsilon, \quad i \in \mathbb{N}^*,\ \epsilon \approx 0 \tag{5.7}$$

5.4 Novos valores de extinção propostos

Verificada a relação entre estrutura Max-tree e determinação de valores de extinção por poda (filtro por atributo), qualquer atributo crescente, associado a um nó, pode ser utilizado como valor de extinção. O Algoritmo 5.1 é proposto, nesse sentido, para a generalização do cálculo de valores de extinção, dado qualquer atributo crescente μ, baseado na informação da Max-tree. Em linhas gerais, a partir de uma folha $\mathcal{N}^L$, inicia-se um caminho em direção à raiz. Quando o pai de um nó $\mathcal{N}^A$ tiver mais de um filho (linha 7), ou seja, for detectada uma ramificação na árvore, verifica-se os irmãos deste nó (linha 8): se o irmão já foi visitado e se há empate de extinção, ou se atributo do irmão for maior (linhas 9-11), então o atributo de $\mathcal{N}^A$ é definido como a extinção de $\mathcal{N}^L$ (linha 19). A Figura 5.4 procede estes passos para uma árvore e atributo μ específicos (número de descendentes). No caminho até a raiz, irmãos do último nó visitado N^A são checados (marcados em amarelo). Se, neste momento, há um irmão N^F com atributo maior, então a extinção da folha de origem passa a ser $\mu(N^A)$ (Fig. 5.4a e 5.4d). Se, em todos os casos, nenhum deles tem atributo maior, então a extinção da folha é "infinita" ou igual ao atributo da raiz (Fig. 5.4b). Se irmão N^F com o mais alto atributo tem o mesmo atributo de N^A e já visitado, então um "empate" é caracterizado e a extinção da folha passa a ser $\mu(N^A)$ (Fig. 5.4c). Finalmente, a imagem é reconstruída (renderizada) com a atribuição de extinção de cada componente de nível $\mathcal{C}_{\mathcal{N}^L}$ de máximo regional representado por um nó folha $\mathcal{N}^L$ (linha 21 do Alg. 5.1). O atributo μ pode ser altura (diferença de nível de cinza), área ou volume, já bem estabelecidos na literatura. Propõe-se agora a definição de novos atributos, associados aos nós da Max-tree (componentes de nível), para que possam ser utilizados como valores de extinção no algoritmo genérico apresentado.

Algoritmo 5.1: Algoritmo genérico para determinação de valores de extinção usando a Max-tree.

ENTRADA: MT_I, μ
SAÍDA: E_μ

EXTINÇÃO()
1 $continua \leftarrow$ **verdadeiro**
2 **para cada** $\mathcal{N}^L \in$ (folhas de MT_I)
3 $extincao \leftarrow \infty$
4 $\mathcal{N}^A \leftarrow \mathcal{N}^L$
5 $\mathcal{N}^P \leftarrow$ pai de $\mathcal{N}^A$
6 **enquanto** $continua$ **e** $\exists \mathcal{N}^P$
7 **se** (número de filhos de $\mathcal{N}^P$) > 1
8 **para cada** $\mathcal{N}^F \in$ (filhos de $\mathcal{N}^P$) **e** $continua$
9 **se** $((\mathcal{N}^F$ já visitado) **e**
10 $\mathcal{N}^F \neq \mathcal{N}^A$ **e** $\mathcal{N}^F_\mu = \mathcal{N}^A_\mu)$
11 **ou** $(\mathcal{N}^F \neq \mathcal{N}^A$ **e** $\mathcal{N}^F_\mu > \mathcal{N}^A_\mu)$
12 $continua \leftarrow$ **falso**
13 $\mathcal{N}^F$ é marcado como visitado
14 **se** $continua$
15 $\mathcal{N}^A \leftarrow \mathcal{N}^P$
16 $\mathcal{N}^P \leftarrow$ pai de $\mathcal{N}^A$
17 $continua \leftarrow$ **verdadeiro**
18 **se** $\exists \mathcal{N}^P$
19 $extincao \leftarrow \mathcal{N}^A_\mu$
20 $\mathcal{N}^L_{ext} \leftarrow extincao$
21 $E_\mu \leftarrow \max\{\mathcal{N}^L_{ext} \cdot \mathcal{C}_{\mathcal{N}^L} \mid \forall \mathcal{N}^L \in$ (folhas de $MT_I)\}$
22 **retorne** E_μ

Como visto no Capítulo 3, algumas medidas podem ser adicionadas incrementalmente em tempo de construção da árvore. Desta forma, a complexidade permanece a mesma do algoritmo original da Max-tree (há solução quase-linear proposta por [36]). No conjunto de informações associadas a cada nó $\mathcal{N}_x$, para novas extinções, destacam-se:

- **Número de descendentes** da subárvore enraizada em $\mathcal{N}_x$ ou, em outras palavras, a cardinalidade do conjunto de nós desta subárvore.

- **Altura topológica** da subárvore enraizada em $\mathcal{N}_x$ ou, de outra forma, a quantidade máxima de arestas no caminho (comprimento máximo do caminho) deste nó raiz até qualquer outro nó descendente (certamente uma folha) nesta subárvore.

- **Altura, largura e diagonal da caixa envolvente** do componente de nível $\mathcal{C}_{\mathcal{N}_x}$, ou diferença entre a maior e menor linha (para altura), entre maior e menor coluna (para largura), e hipotenusa

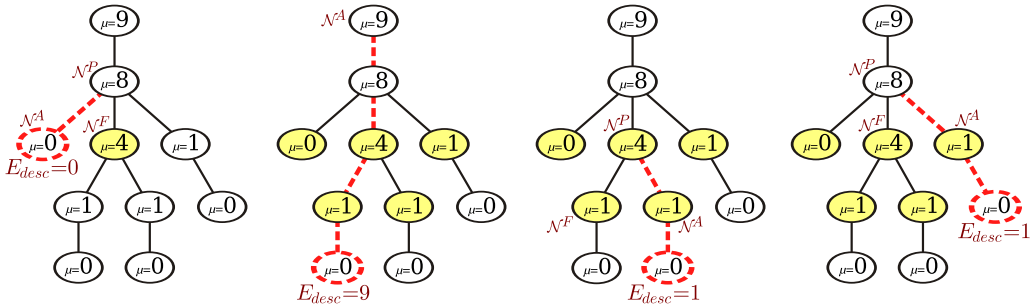


Fig. 5.4: Caminhos a partir de cada folha e verificação de irmãos para cálculo de extinção.

do triângulo retângulo formado de catetos com tamanhos dados por estas duas medidas (para diagonal), supondo todos os *pixels* pertencentes ao componente de nível.

A Figura 5.5 revisa o entendimento destas definições. Por exemplo, o nó C apresenta 6 descendentes (F, G, H, I, J e K) e sua altura topológica é 3, correspondendo ao comprimento máximo de caminho a outro nó descendente (no caso, caminho até K). Em relação à caixa envolvente, por exemplo, o componente de nível, referente a região E, apresenta altura h_E, largura w_E e diagonal $d_E = (h_E^2 + w_E^2)^{1/2}$.

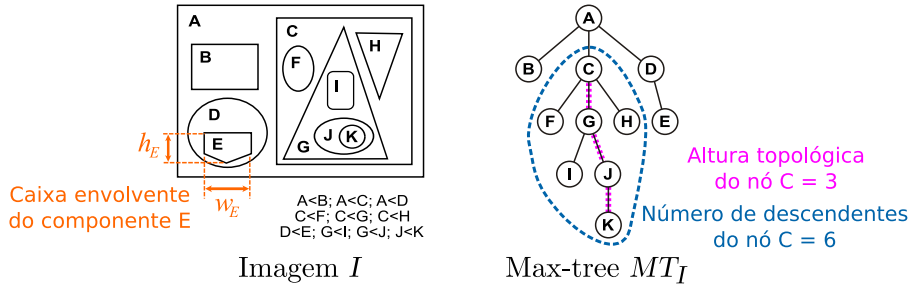


Fig. 5.5: Imagem e sua Max-tree anotadas com os atributos em discussão.

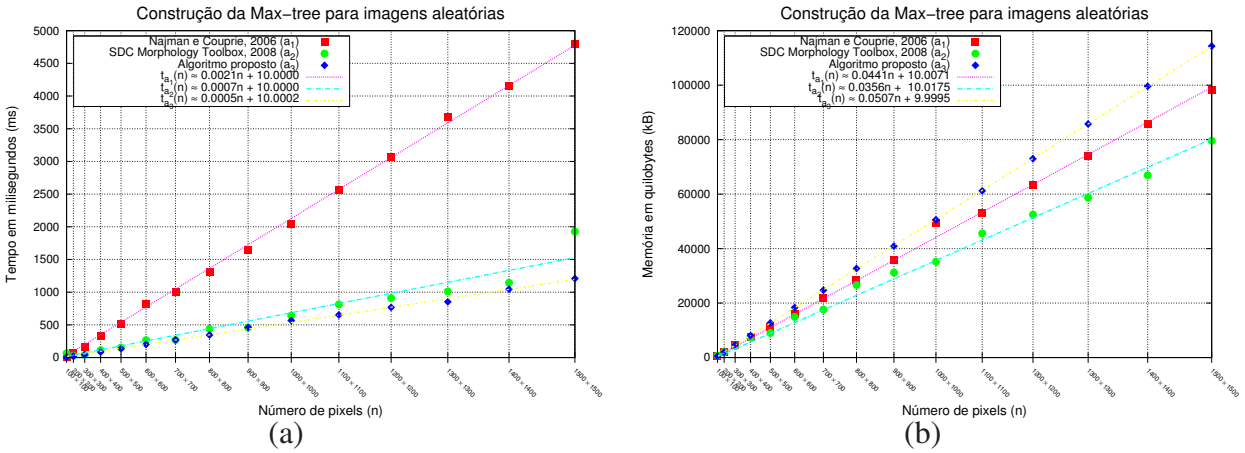
Tais atributos, quando aplicados ao Algoritmo 5.1, definem duas extinções topológicas (no domínio da árvore): descendentes, E_{desc}, e altura da subárvore, E_{htop}; e três extinções geométricas (no domínio da imagem): altura, largura e diagonal da *caixa envolvente*, E_{hbbox}, E_{wbbox} e E_{dbbox}, respectivamente.

A Figura 5.6 compara o tempo de construção da Max-tree (a) e uso de memória (b) para três diferentes algoritmos[5]: a_1 usando *union-find* [36]; a_2 *toolbox*[6] baseada em [1]; a_3 referente aos Algoritmos 3.1 e 3.2 desenvolvidos[7] no Capítulo 3. Os algoritmos a_1, a_2 e a_3 calculam os atributos de altura, área e volume. a_2 e a_3 também determinam a caixa envolvente. O Algoritmo a_3 proposto

[5]Os testes foram feitos em um Mobile Pentium© 4, 3.2GHz, 512MB, e os três algoritmos, implementados em C/C++ (compilação otimizada).

[6]`http://mmorph.com`

[7]`http://www.adessowiki.org` *(na seção* code*)*



(a) (b)

Fig. 5.6: Comparação de algoritmos de Max-tree. (a) Tempo processamento. (b) Consumo de memória.

acrescenta o número de descendentes e a altura topológica incrementalmente. Cada ponto, no gráfico (a), consiste no tempo (em milissegundos) e, no gráfico (b), no consumo de memória (em quilobytes), de uma imagem aleatória I_R com n pixels. $I_R(x) = \text{rand}(L-1), \forall x \in E$, com dimensões $N \times N$ ($n = N^2$ pixels) para $N = 100k$ tal que $k = [1, 15]$ (15 imagens). A implementação proposta tem desempenho equivalente a outras soluções. O cálculo incremental de uma série de novos atributos tem pouca influência no tempo total de construção da árvore. As estruturas de dados utilizadas (vide Cap. 3) possibilitam uma solução relativamente rápida, porém, com a demanda de armazenamento de mais atributos em cada nó, o consumo de memória é ligeiramente mais alto.

Em relação à determinação de extinções, com a Figura 5.7, observa-se a eficiência das cinco propostas implementadas pelo mesmo Algoritmo 5.1. Este depende basicamente do número de folhas (linha 2), da altura topológica da árvore (linha 6) e do número de filhos de cada nó visitado (linha 8). Entretanto, para um mesmo número de nós: **(i)** mais folhas implicam em menor altura da árvore; **(ii)** mais filhos por nó implicam em menor altura da árvore; **(iii)** altura elevada da árvore implica em menos folhas ou menos filhos por nó. Então há um certo balanço entre estas variáveis promovendo uma rápida (tempo mais de três vezes menor que o da construção da Max-tree) e igualitária (todas as extinções determinadas em tempo semelhante) execução como visto na Figura 5.7.

5.4.1 Exemplos

A Figura 5.8 ilustra uma imagem sintética e sua Max-tree com os máximos associados obtidos a partir de atributos tradicionais (altura, área e volume) e os novos valores de extinção propostos neste trabalho (descendentes, altura topológica, altura, largura e diagonal da caixa envolvente ou *bounding box*). A tabela, por sua vez, mostra as extinções de todos os máximos, sendo os maiores realçados

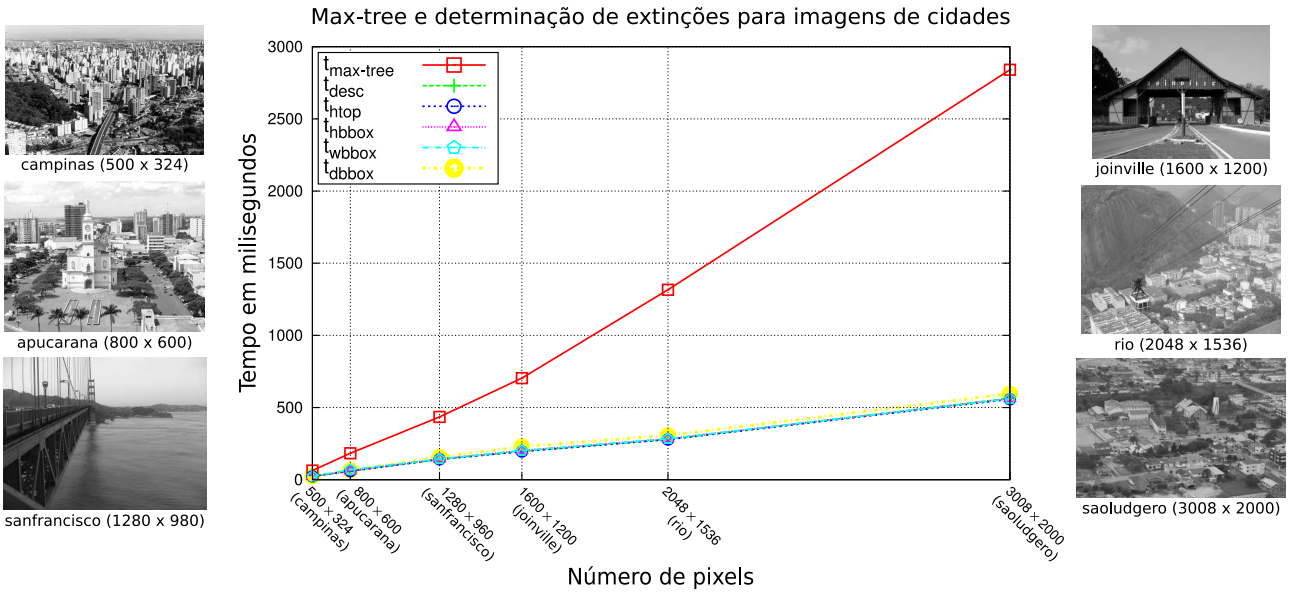


Fig. 5.7: Tempos de construção da Max-tree e de determinação das extinções propostas.

em negrito. Os recursos adicionados sinalizam potenciais aplicações, onde a topologia da árvore ou a geometria do componente pode ser explorada.

A Figura 5.9 mostra algumas imagens reais, onde máximos regionais com extinções relevantes, a partir dos novos atributos, são destacados (com pequena dilatação, em vermelho, para melhor visualização). Somente os máximos regionais, cujos valores de extinções estão acima de um certo valor ou em um intervalo de valores, são selecionados, neste caso, com o propósito de contar objetos como tomates, feijões ou tacos (eventualmente aplicado no sentido de evitar esforço manual repetitivo para esta tarefa). Todos os testes podem ser reproduzidos usando o código-fonte e imagens deste trabalho disponíveis em `http://www.adessowiki.org` na seção "code". Conferência automática de peças a partir de padrões topológicos ou inspeção da integridade de elementos a partir da análise de forma ou tamanho são exemplos de possíveis aplicações auxiliadas por estas extinções diferenciadas.

5.5 Segmentação da Max-tree

Com base na seleção de k folhas da Max-tree (máximos regionais) com as k maiores extinções, pode-se promover a segmentação da árvore e determinar uma região de extinção da imagem. O Algoritmo 5.2 detalha este procedimento. Inicialmente é atribuída extinção, a partir do atributo crescente μ, para cada uma das folhas ou máximos regionais (linha 1). Em seguida, estas folhas são organizadas em ordem decrescente de seus valores de extinção (linha 2). A partir de cada uma das k folhas com maiores extinções (linhas 3 e 4), um percurso é feito a todos os seus ancestrais não visitados no caminho em direção à raiz, e os nós, marcados como visitados e pertencentes à partição da folha

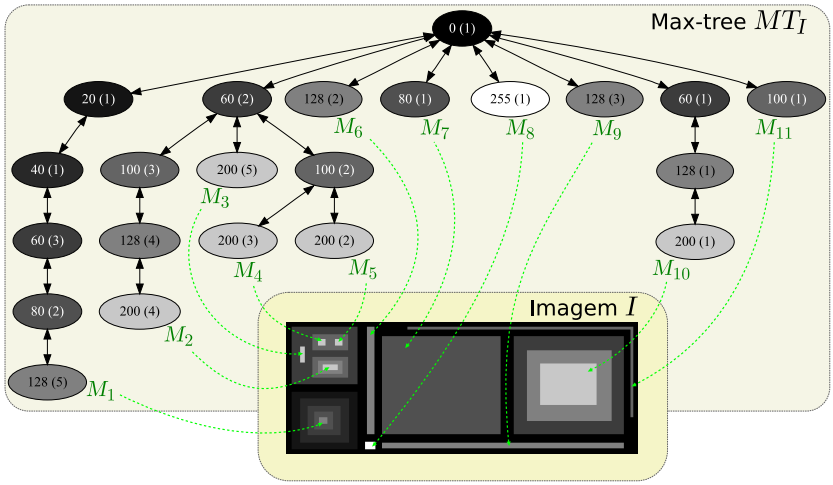


Imagem sintética I e sua Max-tree MT_I com os máximos correspondentes

Nó	E_{altura}	E_{area}	E_{volume}	E_{desc}	E_{htop}	E_{hbbox}	E_{wbbox}	E_{dbbox}
128 (5)	128	4270	166604	5	**6**	61	70	8362
200 (4)	140	4270	355684	**22**	4	61	70	8362
200 (5)	200	85	11900	1	1	17	5	273
200 (3)	140	646	37040	3	2	17	38	1626
200 (2)	100	56	5600	1	1	7	8	86
128 (2)	128	912	116736	1	1	**140**	8	12819
80 (1)	80	**52360**	1048320	1	1	104	126	26235
255 (1)	**255**	88	22440	1	1	8	11	150
128 (3)	128	1542	197376	1	1	6	**374**	65562
200 (1)	200	12168	**3386220**	3	3	104	117	24066
100 (1)	100	999	99900	1	1	96	240	**158451**
Highest	M_8	M_7	M_{10}	M_2	M_1	M_6	M_9	M_{11}

Extinções para cada máximo regional

Fig. 5.8: Ilustração dos valores de extinção para uma imagem sintética I (374×140).

de partida (linhas $5-7$). Caso haja um nó já visitado neste caminho e com partição relativa à outra folha então, deste ponto até a raiz, considera-se como trecho de empate não pertencente à nenhuma partição (linhas $8-10$). Novamente, para cada folha i entre as k de extinções mais relevantes, é feito o caminho em direção à raiz, de tal modo que se detecte o último nó que manteve a partição do ponto de partida (i), ou seja, até que se encontre a raiz da i-ésima subárvore da segmentação da Max-tree (linhas $13-16$). A seleção de máximos relevantes pode ser feita também simplesmente pela escolha de limiares e, neste caso, a ordenação das extinções não se faz necessária. Nota-se que a escolha deste limiar ou de k tem-se mostrado útil na identificação de marcadores internos de objetos na cena (a quantidade de máximos é um resultado imediato para contagem automática). Entretanto, uma caracterização mais precisa (por exemplo, área real ou estimação de volume) destes objetos é ocasionalmente interessante após um processo de segmentação. Os mais variados algoritmos podem

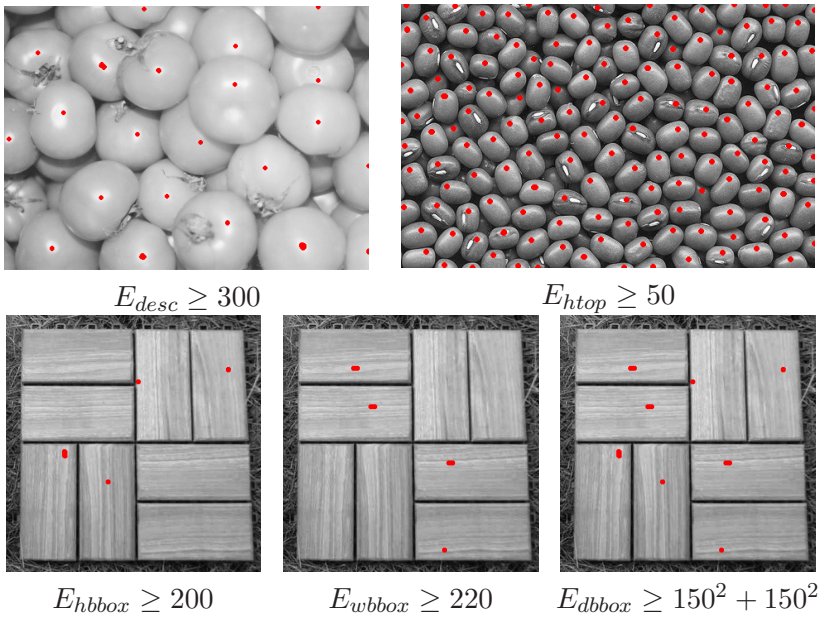


Fig. 5.9: Contagem indireta a partir de valores de extinção.

ser projetados para uma segmentação apropriada dependendo do tipo de imagem de entrada. Nesta seção, a proposta é um método simples baseado na definição de subárvores formadas por zona máxima de influência de folha selecionada (máximo regional com valor de extinção suficientemente alto, por exemplo).

A Figura 5.10 ilustra o procedimento, em detalhes, para $k = 3$ maiores extinções (atributo μ como número de descendentes) de um sinal. Uma vez definidos os máximos, M_A, M_B, ... com as extinções relevantes, caminhos para a raiz, rotulados por A, B, ..., são seguidos. O trecho próximo à raiz com mais de um rótulo recebe a denominação de "caminho concorrente". Após isto, k subárvores são definidas a partir da emanação dos respectivos rótulos a todos os descendentes. A imagem reconstruída (renderizada), a partir desta árvore rotulada, consiste em k componentes conexos, resultando a segmentação em função dos contornos existentes.

5.6 Resultados experimentais

Nesta seção, algumas aplicações de valores de extinção são apresentadas para os novos atributos propostos. Os máximos regionais com significativas extinções são destacados (em vermelho). Ou seja, somente os máximos, cujos valores de extinção estejam acima de um certo valor ou em uma faixa de valores são selecionados. A implementação e imagens são disponibilizadas em `http://www.adessowiki.org`, na seção "code", para reprodução dos testes. Recursos introduzidos neste capí-

Algoritmo 5.2: Algoritmo para segmentação da Max-tree em k subárvores, a partir de extinções mais relevantes.

ENTRADA: MT_I, μ, k
SAÍDA: $MT_{(1)}, MT_{(2)}, \ldots, MT_{(k)}$

SEGMENTAÇÃO_MAX-TREE()
1 EXTINÇÃO(MT_I,μ) *//vide Algoritmo 5.1*
2 $elemento \leftarrow$ ordenação das folhas de MT_I por valores decrescentes de extinção de μ
3 **para cada** $i \in \{1, 2, \ldots, k\}$
4 $\mathcal{N} \leftarrow elemento[i]$
5 **enquanto** $\exists \mathcal{N}$ **e** ($\mathcal{N}$ não visitado)
6 $\mathcal{N}$ é marcado como visitado e pertencente à partição i
7 $\mathcal{N} \leftarrow$ pai de $\mathcal{N}$
8 **enquanto** $\exists \mathcal{N}$ **e** ($\mathcal{N}$ já visitado) **e** (partição de $\mathcal{N} \neq 0$) *//se trecho até raiz já percorrido*
9 $\mathcal{N}$ é marcado com partição 0 *//zona de empate não deve pertencer à nenhuma partição*
10 $\mathcal{N} \leftarrow$ pai de $\mathcal{N}$
11 **para cada** $i \in \{1, 2, \ldots, k\}$
12 $\mathcal{N} \leftarrow elemento[i]$
13 **enquanto** $\exists \mathcal{N}$ **e** (partição de $\mathcal{N} = i$)
14 $\mathcal{N}^A \leftarrow \mathcal{N}$
15 $\mathcal{N} \leftarrow$ pai de $\mathcal{N}$
16 $MT_{(i)} \leftarrow \mathcal{N}^A$
17 **retorne** $MT_{(1)}, MT_{(2)}, \ldots, MT_{(k)}$

tulo podem ser usados em sistemas de visão para evitar esforço de contagem manual de peças, objetos, frutas, células, etc, com maior rapidez e, em alguns casos, com maior precisão, supondo inexistência de interferências como fadiga, por exemplo. Neste sentido, a Figura 5.11 mostra alguns exemplos em que as extinções significativas de descendentes e altura topológica se adaptam para a contagem de feijões, brigadeiros e células sanguíneas (estes dois últimos com extinções da imagem negativa). Outra aplicação de visão é a de controle de processos industriais como, por exemplo, verificação da integridade de rótulos, da presença de todas as peças de um conjunto, da inspeção automática de formas, tamanhos, quebras, riscos, cores, entre outras. A última coluna da Figura 5.11 exibe as extinções significativas (em um dado intervalo) de largura de caixa envolvente de comprimidos, com aplicação na indústria farmacêutica.

5.6.1 Análise

Uma importante característica dos valores de extinção é a imunidade a ruídos. Na Figura 5.12, ruído Gaussiano, com diferentes variâncias σ^2, é acrescentado na imagem de brigadeiros. Extinções de descendentes (a) e de diagonal da caixa envolvente (b) são aplicadas para contagem dos doces e, em todos os casos, há pouco distanciamento dos máximos relevantes selecionados, demonstrando a

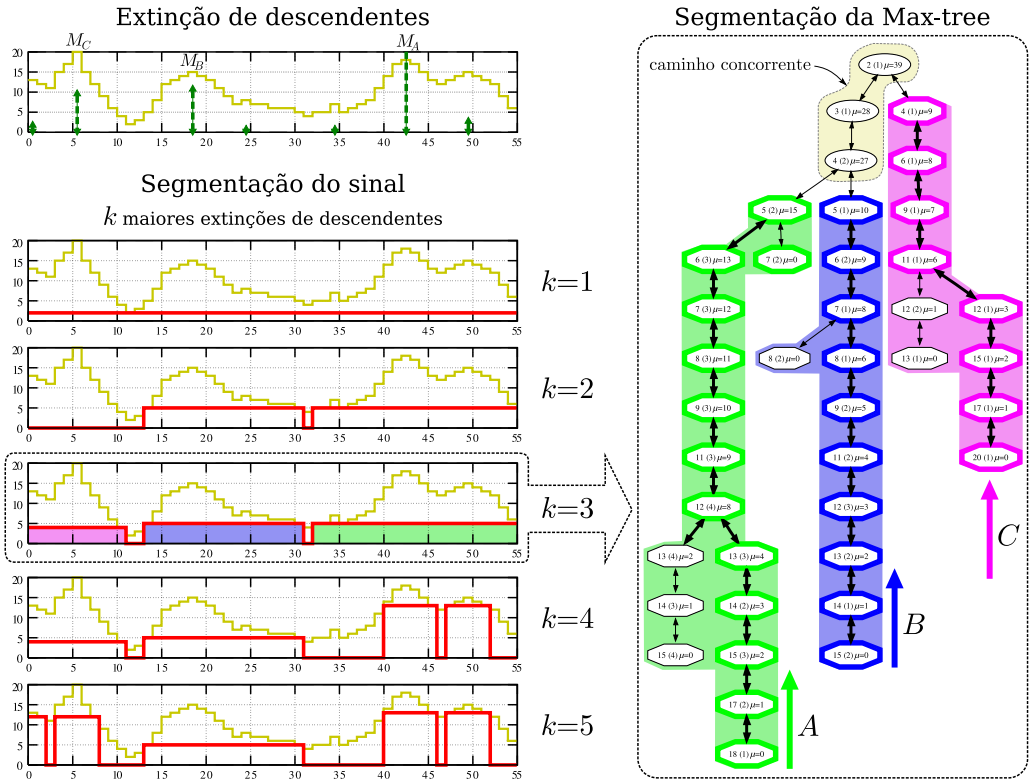


Fig. 5.10: Segmentação da Max-tree.

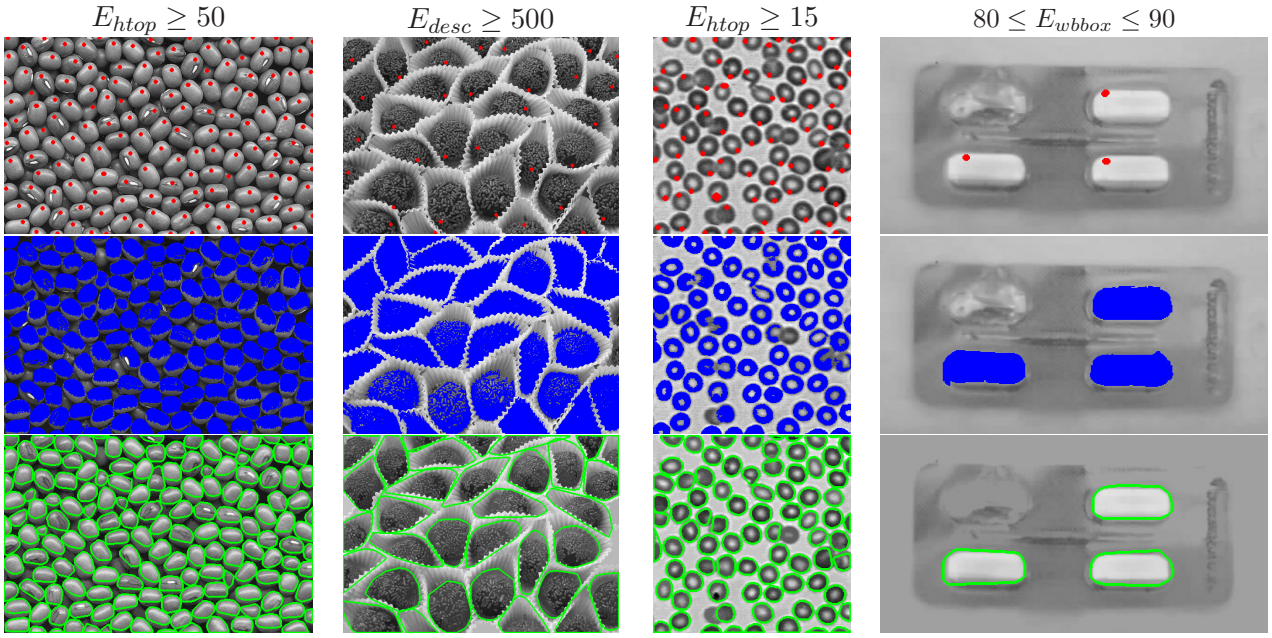
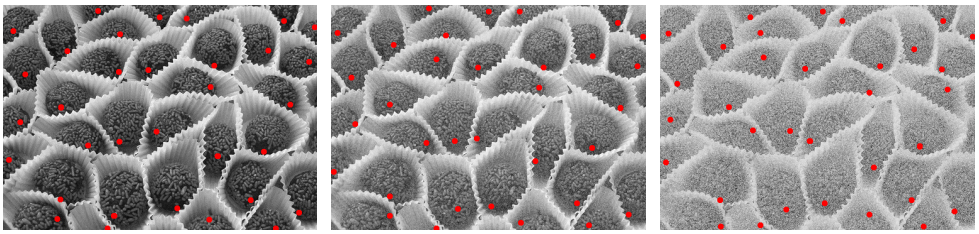


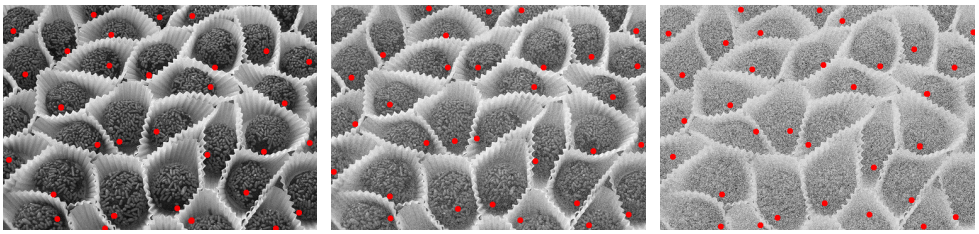
Fig. 5.11: Exemplos de segmentação. Máximos regionais selecionados (acima). Regiões de extinção (centro). Reconstrução da imagem e fecho convexo dos componentes (abaixo)

robustez mencionada.



$original$ $\sigma^2 = 0,01$ $\sigma^2 = 0,1$

(a) $E_{desc} \geq 500$



$original$ $\sigma^2 = 0,01$ $\sigma^2 = 0,1$

(b) $E_{dbbox} \geq 50^2 + 50^2$

Fig. 5.12: Robustez a ruído Gaussiano.

Outra questão importante é o estabelecimento de um paralelo entre um conjunto clássico de operadores morfológicos [38] e a segmentação da Max-tree proposta, no sentido de ilustrar o ganho que se pode ter na obtenção de uma solução. A Figura 5.13 exemplifica esta comparação para uma imagem de ladrilho. Observa-se, neste caso, que a técnica tradicional necessita de uma quantidade maior de etapas e de operadores relativamente mais custosos computacionalmente para concretizar uma boa segmentação dos retângulos maiores.

5.7 Complexidade

Para a determinação de um valor de extinção qualquer, com base no Algoritmo 5.1 desenvolvido, é necessário visitar todas as folhas inicialmente (da ordem de N em pior caso). A partir de cada folha, caminha-se em direção à raiz (L passos no pior caso). Neste percurso, a cada nó visitado, verificam-se os atributos de todos os irmãos (da ordem de N em pior caso). Portanto, a complexidade do cálculo de extinção é $O(N^2 \times L)$. Supondo $L = 256$ para imagens usuais de 8 bits, o algoritmo é, na prática, quadrático. No que se refere à técnica de segmentação da Max-tree, feita pelo Algoritmo 5.2, o primeiro passo exigido é o cálculo da extinção, segundo o atributo crescente μ escolhido, em tempo quadrático. Feito isto, faz-se necessária a ordenação das folhas pelos seus valores de extinção (da ordem de $N \log N$). Para cada folha com k maior extinção, caminha-se até a raiz (trecho,

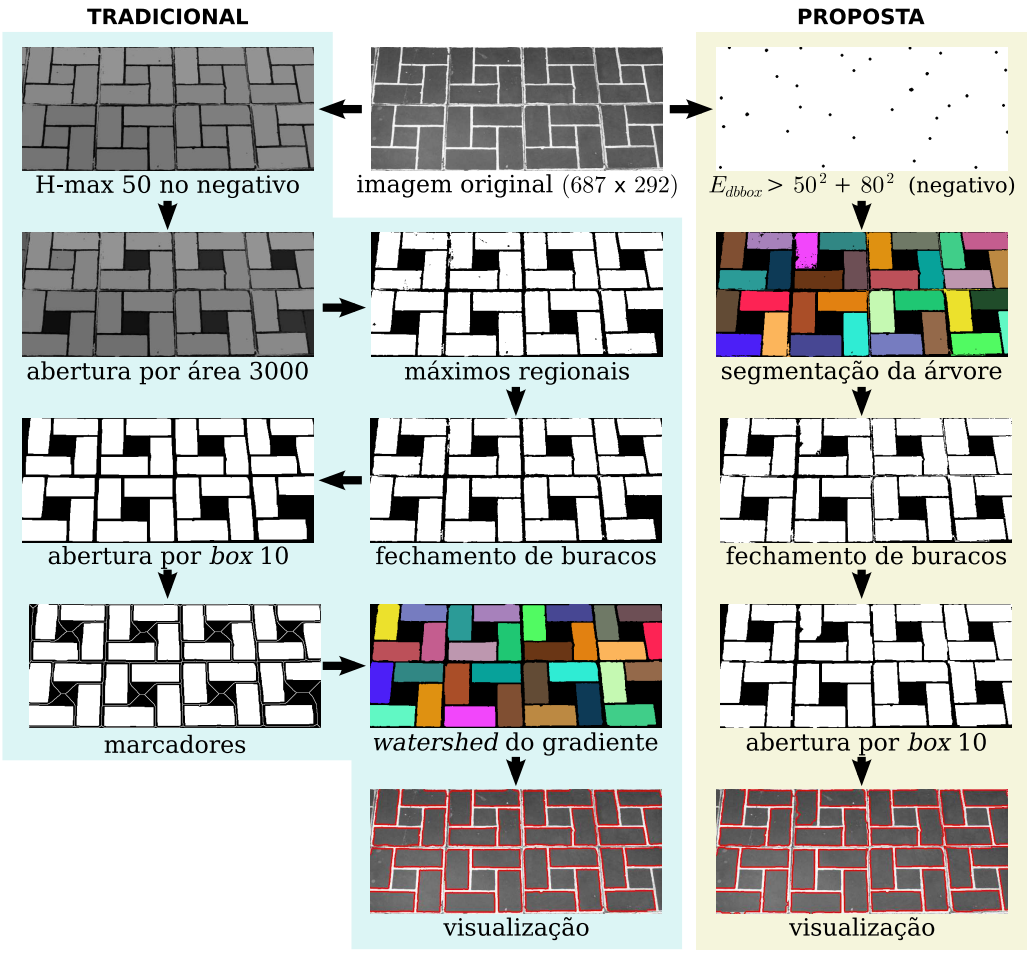


Fig. 5.13: Comparação da segmentação da árvore com um método tradicional.

portanto, proporcional a $k \times L$). Este último procedimento se repete ainda uma vez para a definição das subárvores geradas (também em $k \times L$). Desta forma, a segmentação da árvore é da ordem $N^2 \times L + N \log N + 2k \times L$, ou seja, de complexidade $O(N^2 \times L)$, supondo k com valor limitado em N (pode-se quebrar a árvore em, no máximo, a quantidade existente de nós). A Tabela 5.1 resume estas informações.

Tab. 5.1: Complexidade do algoritmo genérico para cálculo de valores de extinção e da segmentação da Max-tree em k subárvores, sendo N o número de nós e L a quantidade de níveis de cinza.

Algoritmo	Complexidade
EXTINÇÃO	$O(N^2 \times L)$
SEGMENTAÇÃO_MAX-TREE	$O(N^2 \times L)$

5.8 Considerações sobre o capítulo

Os valores de extinção são ferramentas poderosas em aplicações de visão computacional, indicando, com um esforço relativamente reduzido de processamento, informações de posicionamento de objetos, segundo suas características topológicas ou geométricas. Mesmo com imagens ruidosas e ampla variação da faixa de valores de extinção, os resultados da seleção de máximos regionais com extinções destacadas se mantêm robustas. E apenas a seleção de extinção por faixas (limiarizações) foi suficiente para detectar e contar objetos de interesse nas experimentações. A implementação de Max-tree é baseada em [1], modificada para calcular os atributos incrementalmente. A eficiência de execução e consumo de memória são equivalentes (ou similares) a outros algoritmos, mesmo com o cálculo de novos atributos (como número de descendentes, altura topológica e cantos da caixa envolvente). Cinco novos valores de extinção foram propostos. A extinção de descendentes sinaliza a complexidade de uma região, no sentido de haver muitas pequenas elevações e cumes na parte do relevo associada, sendo útil na detecção de texturas. A altura topológica indica a quantidade de "camadas de terra" sobrepostas, sendo útil na detecção de objetos esféricos ou piramidais. Finalmente, altura, largura e diagonal da caixa envolvente refletem as dimensões aproximadas, em pixels, de um objeto (mais claro que o fundo) na imagem. Além disto, um método simples de segmentação, baseado nas extinções mais relevantes, é apresentado. Obviamente este não se aplica a qualquer tipo de imagem. Bons resultados ocorrem quando há uma separação nítida entre objetos de interesse (por exemplo, a imagem original na Fig. 5.13). Procedimentos usuais de realces de bordas podem contribuir com estas separações. Diferentes maneiras de seleção de extremos relevantes, baseadas nos cinco novos critérios de extinção, foram apresentadas. Como aplicação direta, pode-se citar o particionamento e simplificação de imagens, favorecendo, por sua vez, passos subsequentes de processamento, como a detecção de formas desenvolvida no próximo capítulo.

Capítulo 6

Detecção de formas

A determinação de formas em imagens tem significativa importância em diversas aplicações: vetorização de sistemas de informação geográfico (GIS) [109], sensoriamento remoto, desenhos de circuitos eletrônicos, de plantas baixas, mecânicos [110] ou artísticos, segmentação e interpretação de referências para assistência à direção de veículos [111] ou automatização de trajetórias de robôs, análise de estruturas discriminantes de imagens biométricas como as minúcias nas impressões digitais, reconhecimento de caracteres (OCR) ou outros símbolos, como notações musicais (OMR) [112], logotipos [113], informações em etiquetas, cheques ou formulários [114], entre outros inúmeros exemplos. A transformada de Hough [115] é um interessante e computacionalmente eficiente – se o lugar geométrico de busca for modelado com quantidade suficientemente pequena de parâmetros – procedimento para detecção de retas, círculos e elipses [116] e tem sido amplamente usada para resolver problemas de segmentação de formas geométricas parametrizáveis em imagens binárias. Entretanto, detecções podem ser feitas diretamente em imagens em níveis de cinza como, por exemplo, de elementos mais simples como retas [117] ou outros mais complexos apoiados por treinamento e minimização de energia [118].

Este capítulo propõe um algoritmo genérico para detecção de objetos contrastantes de interesse baseado na caracterização dos componentes da Max-tree. A Seção 6.1 ilustra o algoritmo genérico proposto e a modelagem de algumas formas para busca e detecção. Na Seção 6.2, alguns experimentos e indicações de tempos relativos de execução são apresentados. Por fim, considerações sobre o assunto são feitas na Seção 6.3.

6.1 Proposta de detecção de formas

Esta seção consiste no desenvolvimento de um algoritmo genérico para detecção de formas a partir de regiões da Max-tree, sendo o texto baseado em publicações efetuadas sobre o tema [29, 119, 30].

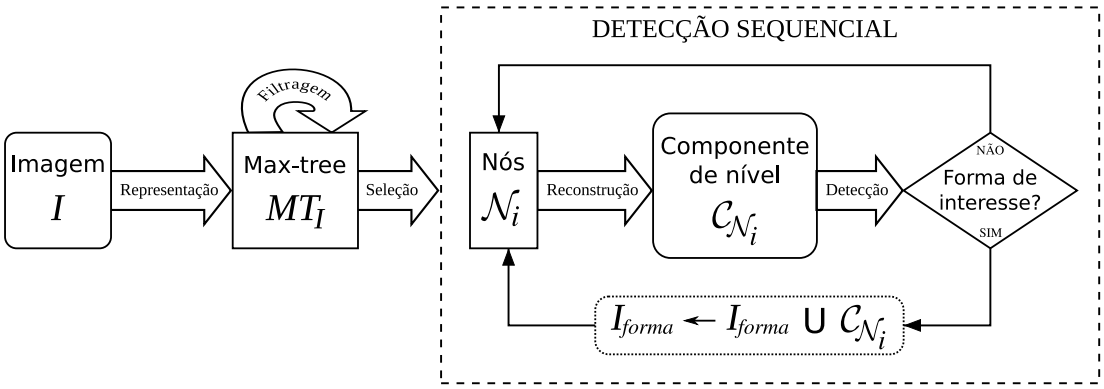


Fig. 6.1: Estágios da detecção de formas.

O diagrama da Figura 6.1 ilustra alguns passos relacionados a este processo. Após a construção de MT_I de uma imagem I, a ideia é buscar nós que representem componentes conexos de interesse. Para um desempenho melhor do método, eventualmente pode-se estabelecer uma filtragem para exclusão de nós, cujos atributos extrapolam o objetivo de uma aplicação (por exemplo, a remoção de componentes com área próxima a um pixel). Para cada nó $\mathcal{N}$ da árvore remanescente após a filtragem (se for o caso), obtém-se a imagem binária do componente conexo associado por meio da reconstrução que resulta $\mathcal{C}_{\mathcal{N}}$, conforme descrito no Algoritmo 3.3 (Cap. 3). Este procedimento refere-se a um crescimento de regiões, limitado aos níveis de cinza da imagem maiores que o nível do nó $\mathcal{N}_{nivel}$, a partir da coordenada de semente $\mathcal{N}_x$ registrada, quando na formação da Max-tree. A imagem binária $\mathcal{C}_{\mathcal{N}}$, contendo um único componente conexo, é analisada por uma função "DETECTA()", projetada no sentido de verificar se a disposição espacial de um conjunto de pixels (com intensidade 1 na imagem binária) caracteriza ou não o padrão de interesse. Em caso afirmativo, $\mathcal{C}_{\mathcal{N}}$ é agregado à uma imagem resultante I_{forma}. Evidentemente que no domínio discreto da imagem, alguma tolerância de posicionamento dos pixels deva ser levada em consideração, sobretudo na tentativa de reconhecer um lugar geométrico (por exemplo, com arranjo próximo à descrição de uma circunferência ou elipse). Desta maneira, valores $d_{\min}$ e $d_{\max}$ são arbitrariamente estipulados como intervalo de tolerância na análise de um componente de nível, podendo assumir diferentes significados, conforme o parâmetro principal (ou combinação de parâmetros) de decisão adotado (por exemplo, diferenças máximas entre distâncias, raios, espessuras, entre outros).

A visitação dos nós, para a análise da forma associada aos seus pixels, segue ordem dada por um percurso em profundidade. Esta escolha contribui com o desempenho pelo fato da não necessidade, em alguns casos, de tratar os descendentes de nós selecionados por terem uma relação de inclusão de componentes (conforme visto na Sec. 3.1). Por exemplo, se $\mathcal{C}_{\mathcal{N}}$ representa uma linha com espessura d de interesse, então não há sentido pesquisar regiões que estão incluídas neste, por dois motivos: **(i)** mesmo que haja descendente com espessura na faixa de tolerância, a imagem binária resultante será

a mesma; **(ii)** qualquer descendente terá espessura menor, podendo inclusive estar fora da tolerância. Portanto, uma vez detectado um componente de interesse, seus descendentes são ignorados e análises de formas – que eventualmente podem apresentar custo computacional elevado – são evitadas. Finalizado o percurso, tem-se uma imagem com todos os componentes selecionados, ou seja, com a detecção da forma procurada em todas as ocorrências da mesma no domínio da imagem (se houver). O Algoritmo 6.1 implementa este método. Como entrada, são fornecidas a imagem em níveis de cinza I, a faixa de tolerância entre $d_{\min}$ e $d_{\max}$, e a função "DETECTA()" para a análise do padrão a ser reconhecido. Esta, por sua vez, recebe como entrada uma imagem binária com um único componente conexo. Como saída, é produzida I_{formas} com todas as detecções promovidas na verificação dos componentes da Max-tree. Estes são visitados em profundidade com auxílio de uma pilha de nós. O nó raiz é empilhado inicialmente (linha 4) e um laço, definido enquanto a pilha contiver algum elemento para repetir o procedimento de desempilhar um nó $\mathcal{N}$ (linha 6), verificar a forma de $\mathcal{C}_{\mathcal{N}}$ (linha 7), se de interesse, unir à imagem resultante (linhas 7 e 8), caso contrário (linha 9), empilhar os filhos de $\mathcal{N}$ (linha 10) para continuar a consulta aos descendentes.

Algoritmo 6.1: Algoritmo para detecção de formas.

ENTRADA: I, $d_{\min}$, $d_{\max}$, DETECTA()
SAÍDA: I_{forma}

INICIALIZAÇÃO:
1 $I_{forma}[x] \leftarrow 0, \forall x \in E \subset \mathbb{N}^2$
2 $\mathcal{N}^R \leftarrow$ raiz de MT_I
3 $pilha \leftarrow \emptyset$

DETECÇÃO_DE_FORMAS
4 $pilha$.push$(\mathcal{N}^R)$
5 **enquanto** $pilha \neq \emptyset$
6 $\quad \mathcal{N} \leftarrow pilha$.pop()
7 $\quad$ **se** DETECTA $(\mathcal{C}_{\mathcal{N}}, d_{\min}, d_{\max})$
8 $\quad\quad I_{forma} \leftarrow I_{forma} \cup \mathcal{C}_{\mathcal{N}}$
9 $\quad$ **senão**
10 $\quad\quad pilha$.push(filhos de $\mathcal{N}$)
11 **retorne** I_{forma}

O estágio de reconhecimento da Figura 6.1, como mencionado, é implementado por "DETECTA()" do Algoritmo 6.1. Esta função pode ser implementada basicamente de três maneiras: **(i) por análise geométrica** a partir da descrição matemática do lugar geométrico de figuras parametrizáveis como, por exemplo, círculos, retas, arcos, elipses, entre outras curvas, etc; **(ii) por análise morfológica**, ou seja, com alguma propriedade de forma que se possa explorar como, por exemplo, máximo da trasformada distância para restrição de espessura e detecção de linhas, estrutura peculiar de objetos

conforme um padrão *hit or miss* [38], entre outras técnicas auxiliares como transformada de abertura [38] para espectro de padrões, afinamento, fecho convexo, apenas para citar algumas; **(iii) por casamento de modelo** para objetos conhecidos, descritos por meio de *templates* binários, que devem ser casados com cada componente da árvore, ideal para símbolos, com equacionamento relativamente complexo como, por exemplo, caracteres, ideogramas, logotipos, etc. Este trabalho se concentra em formas geométricas simples para demonstrar o algoritmo. Um "DETECTA()" é implementado para cada padrão (retornando sempre "verdadeiro", se forma encontrada, ou "falso", caso contrário), consistindo na verificação de componentes com base na morfologia ou lugar geométrico de seus pontos no espaço. O significado da tolerância $d_{\min}$ e $d_{\max}$, para cada forma tratada, é apresentado no restante desta seção. As detecções de linhas usando morfologia matemática, e de formas parametrizáveis – como retas, círculos, arcos e elipses – usando equações da geometria espacial, são desenvolvidas a seguir.

6.1.1 Linhas

Um componente conexo de um nó $\mathcal{C}_\mathcal{N}$ é uma linha se todos os pixels pertencem ao contorno $\Psi(\mathcal{C}_\mathcal{N})$ ou se o máximo de sua transformada de distância é igual a 1 [29]. A Equação 6.1 considera certa tolerância para este valor e, consequentemente, implementa a detecção de linhas com certa espessura.

$$espessura_{\min} \leq \max(\mathcal{D}(\mathcal{C}_\mathcal{N})) \leq espessura_{\max} \tag{6.1}$$

A Figura 6.2(a) esquematiza esta aproximação. A detecção de linhas é usualmente aplicada sobre uma imagem pré-processada por um filtro passa-alta ou operador de gradiente para realce de bordas. O Algoritmo 6.2 propõe um refinamento na detecção, quando há uma linha mas o critério mencionado falha. Basicamente é procedida uma dilatação – por $\mathcal{E}_{caixa^{(d_{max})}}$, elemento estruturante com caixa de raio d_{max} –, dilatação da subtração deste conteúdo e intersecção com a imagem original. Supondo o $\mathcal{C}_\mathcal{N}$ da Figura 6.3(a), sua transformada de distância corresponde à Figura 6.3(b). Sendo o máximo valor de $\mathcal{D}(\mathcal{C}_\mathcal{N})$ fora do intervalo de tolerância, é necessário eliminar parte da região, cuja transformada de distância seja maior que o valor d_{max} pré-definido. A Figura 6.3(c) mostra o resultado da execução do algoritmo de refinamento.

Algoritmo 6.2: Algoritmo para melhorar a detecção de linhas em componentes conexos com máximo da transformada de distância maior que d_{max}.

ENTRADA: $\mathcal{C}_{\mathcal{N}}, d_{max}$
SAÍDA: I_{linhas}

REFINAMENTO
1 $aux \leftarrow Dil_{\mathcal{E}_{caixa(d_{max})}}(\mathcal{D}_{\mathcal{E}_{cruz}}(\mathcal{C}_{\mathcal{N}}) > d_{max})$
2 $I_{linhas} \leftarrow \mathcal{C}_{\mathcal{N}} \cap Dil_{\mathcal{E}_{cruz}}(\mathcal{C}_{\mathcal{N}} - aux)$
3 **retorne** I_{linhas}

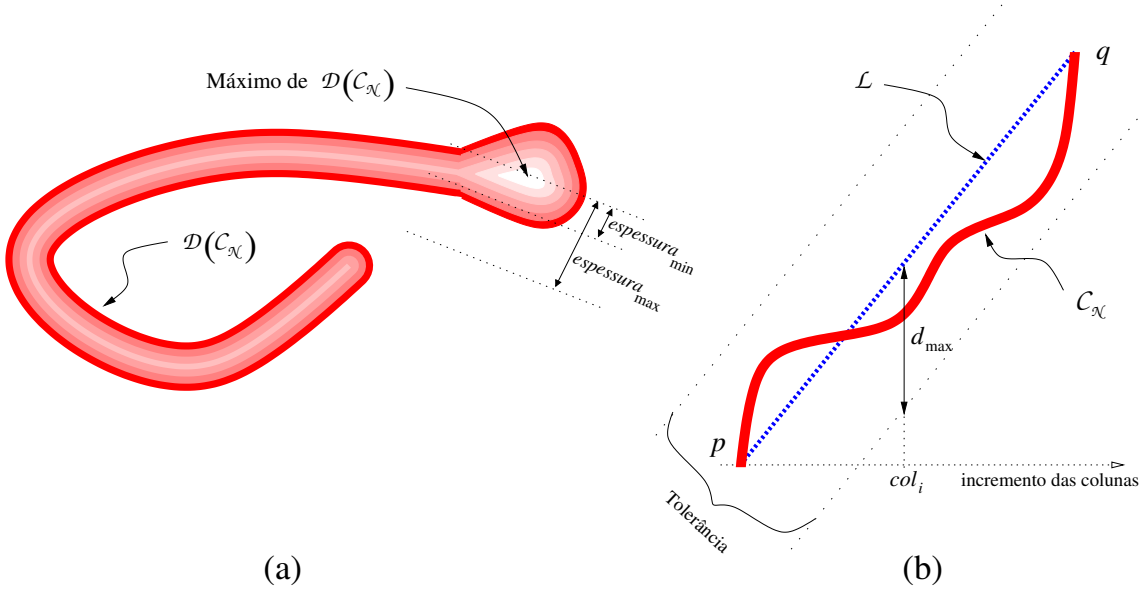


Fig. 6.2: Aproximação de reconhecimento. (a) Linha. (b) Reta.

Fig. 6.3: Refinamento da detecção de linhas. (a) Componente conxexo $\mathcal{C}_{\mathcal{N}}$. (b) Transformada de distância, onde $\max(\mathcal{D}(\mathcal{C}_{\mathcal{N}})) = 70$. (c) Supressão das regiões onde $\mathcal{D}(\mathcal{C}_{\mathcal{N}}) > d_{max}$.

6.1.2 Retas

Considerando dois pixels, p e q, diferentes de zero de um componente conexo $\mathcal{C}_{\mathcal{N}}$, onde o primeiro corresponde ao menor valor de coluna (ou linha) – mais à esquerda (ou acima) – e o outro, ao maior valor de coluna (ou linha) – mais à direita (ou abaixo), uma linha digital $\mathcal{L}$ entre eles, com coeficiente angular $(p_{col} - q_{col})/(p_{lin} - q_{lin})$, é traçada (caso o denominador seja zero, trata-se de uma reta vertical

a ser tratada separadamente). Se $|p_{col} - q_{col}| > |p_{lin} - q_{lin}|$, então as colunas de p até q são aumentadas, caso contrário, as linhas são aumentadas conforme o algoritmo de [120]. Para cada incremento, o valor do pixel em $\mathcal{L}$ é comparado com o valor dos pixels correspondentes em $\mathcal{C}_\mathcal{N}$. Se esta distância está entre $d_{\min}$ e $d_{\max}$ para todos os pixels de $\mathcal{C}_\mathcal{N}$, então $\mathcal{C}_\mathcal{N}$ é uma reta. $d_{\min}$ é normalmente 0. $d_{\max}$ é igual para ambos lados de $\mathcal{L}$ e está relacionado com a tolerância de distância entre o pixel de $\mathcal{C}_\mathcal{N}$ e $\mathcal{L}$. As Equações 6.2 sintetizam esta ideia. A Figura 6.2(b) exibe esta eficiente aproximação dada pela subtração simples em vez do cálculo de distância euclidiana.

$$\begin{aligned} d_{\min} \leq |\mathcal{L}_{col} - p(\mathcal{C}_\mathcal{N})_{col}| \leq d_{\max} \quad &\text{se } |p_{col} - q_{col}| > |p_{lin} - q_{lin}| \\ d_{\min} \leq |\mathcal{L}_{lin} - p(\mathcal{C}_\mathcal{N})_{lin}| \leq d_{\max} \quad &\text{se } |p_{col} - q_{col}| \leq |p_{lin} - q_{lin}| \end{aligned} \tag{6.2}$$

6.1.3 Círculos

Um componente conexo é um círculo ou circunferência se todos os pixels x_i de seu contorno $\Psi(\mathcal{C}_\mathcal{N})$ têm a mesma distância até o pixel centroide x_c. Ao considerar certa tolerância, a Equação 6.3 pode ser descrita.

$$raio^2_{\min} \leq \text{dist}(x_i, x_c)^2 \leq raio^2_{\max}\,, \quad \forall i \in [1, b] \tag{6.3}$$

Onde $b = \text{área}(\Psi(\mathcal{C}_\mathcal{N}))$ ou número de pixels da borda. A Figura 6.4(a) exibe esta aproximação.

6.1.4 Arcos

Um componente conexo de um nó $\mathcal{C}_\mathcal{N}$ pode ser considerado um arco. Para isto, o centroide x_c de $\mathcal{C}_\mathcal{N}$ é calculado. O pixel mais próximo de $\mathcal{C}_\mathcal{N}$ até x_c é selecionado. Um disco (de tamanho proporcional ao $raio_{\max}$ é escolhido) ao redor deste é definido. Um novo centroide y_c de todos os pixels neste disco é calculado. Um segmento de reta é então a ser definido a partir de y_c passando por x_c. Em algum ponto z_{alvo} sobre este segmento, um pixel equidistante de todos os pixels da borda $\Psi(\mathcal{C}_\mathcal{N})$ pode existir. A busca por esta coordenada é implementada considerando a tolerância de raios entre $raio_{\min}$ e $raio_{\max}$ conforme a Equação 6.4. A Figura 6.4 ilustra esta proposta.

$$raio^2_{\min} \leq \text{dist}(x_i, z_{alvo})^2 \leq raio^2_{\max}\,, \quad \forall i \in [1, b]\,, \quad \exists z_{alvo} \in \overrightarrow{y_c x_c} \tag{6.4}$$

Onde $b = \text{área}(\Psi(\mathcal{C}_\mathcal{N}))$ ou número de pixels da borda. A Figura 6.4(b) representa estas conside-

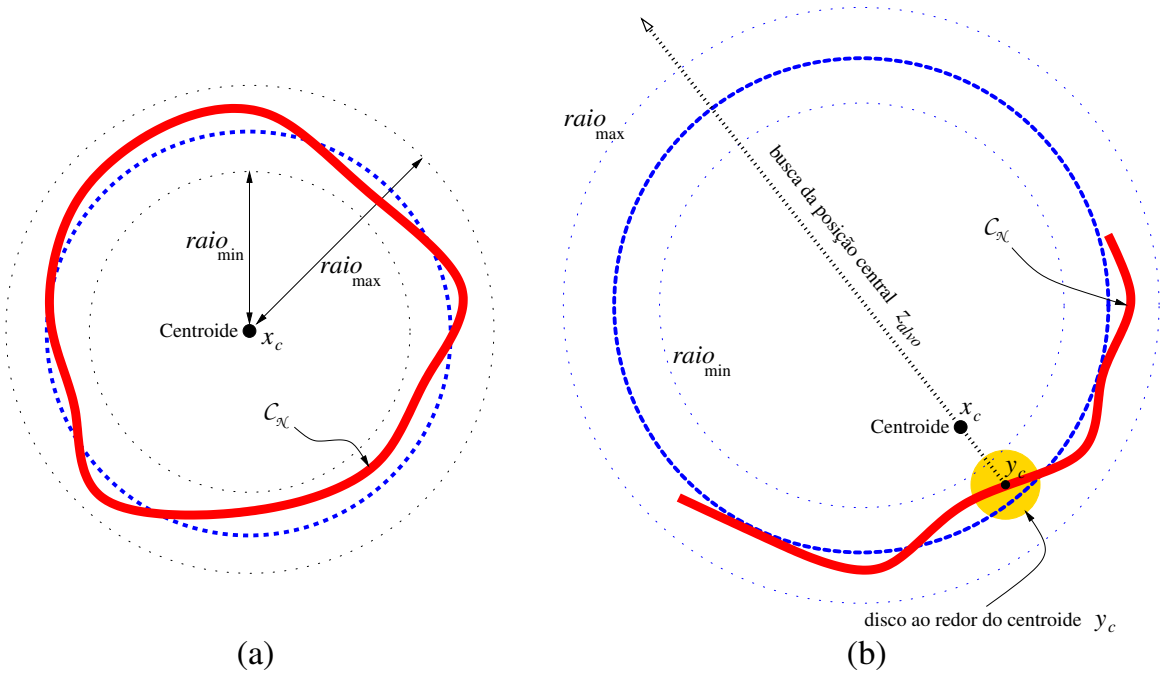


Fig. 6.4: Aproximação de reconhecimento. (a) Círculo. (b) Arco.

rações.

6.1.5 Elipses

Por fim, um componente conexo de um nó $\mathcal{C}_\mathcal{N}$ pode ser uma elipse. Supondo a ocorrência desta figura geométrica, calcula-se primeiramente o centroide x_c de $\mathcal{C}_\mathcal{N}$. O pixel mais afastado x_a e o mais próximo x_b de $\mathcal{C}_\mathcal{N}$ até x_c são selecionados, conforme a Figura 6.5. Com isto, caso trate-se de uma elipse, o raio maior a e menor b são definidos, assim como $c = (a^2 + b^2)^{1/2}$, utilizado na relação $dist(x_{f_1}, x_{f_2}) = 2c$. As coordenadas do ponto focal x_{f_1} são determinadas pelas proporções $x_{f_{1_{lin}}} = c(x_{a_{lin}} - x_{c_{lin}})/a + x_{c_{lin}}$ e $x_{f_{1_{col}}} = c(x_{a_{col}} - x_{c_{col}})/a + x_{c_{col}}$. Assim como as de x_{f_2} dadas por $x_{f_{2_{lin}}} = x_{c_{lin}} + (x_{c_{lin}} - x_{f_{1_{lin}}})$ e $x_{f_{2_{col}}} = x_{c_{col}} + (x_{c_{col}} - x_{f_{1_{col}}})$. Decide-se afirmativamente por uma elipse se todos os pixels x_i da borda obedecerem a Equação 6.5, ou seja, se houver tolerância na igualdade de definição deste lugar geométrico: $dist(x_i, x_{f_1}) + dist(x_i, x_{f_2}) \approx 2a$, para todos os pontos x_i do contorno do componente.

$$d_{\min} \leq (dist(x_{f_1}, x_i) + dist(x_{f_2}, x_i) - 2a) \leq d_{\max}\,, \quad \forall i \in [1, \text{área}(\Psi(\mathcal{C}_\mathcal{N}))] \tag{6.5}$$

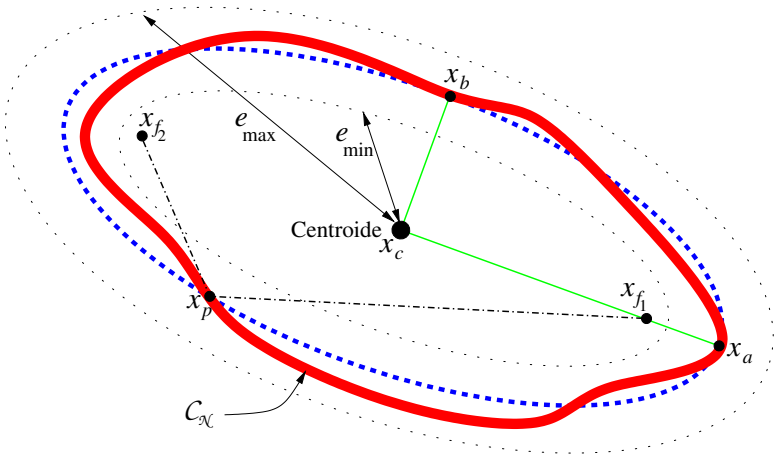


Fig. 6.5: Aproximação de reconhecimento de elipse.

Exemplos simples

A Figura 6.6 resume a aplicação destas aproximações em uma imagem binária sintética contendo linhas, reta, círculos, arco e elipses. Imagens fotográficas são usadas na próxima seção.

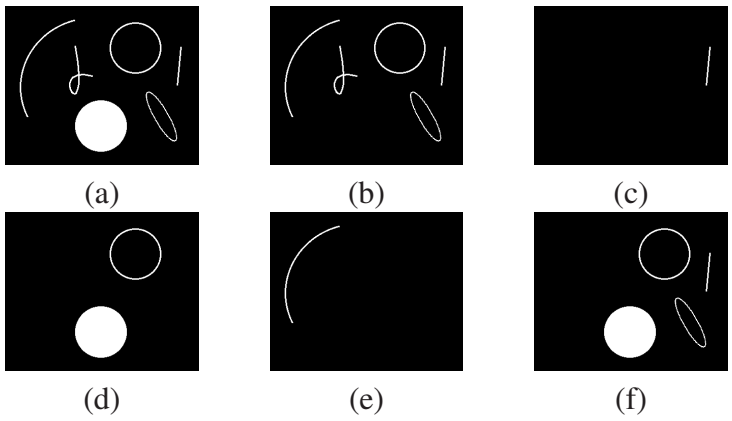


Fig. 6.6: Exemplos de reconhecimento para as aproximações apresentadas. (a) Imagem sintética original I. (b) Linhas em I. (c) Reta em I. (d) Círculos em I. (e) Arco em I. (f) Elipses em I.

6.2 Experimentos

Nesta seção, um conjunto de testes é feito para demonstrar a proposta de reconhecimento de formas, em imagens em níveis de cinza, deste trabalho. Notações são usadas ao lado das figuras para cada imagem i: p_i é o número total de pixels; N_i é o número total de nós (ou regiões) da Max-tree; n_i é o número total de nós selecionados (ou regiões que representam uma forma geométrica

específica); T_i é o tempo total, em milissegundos, de execução (incluindo a construção da Max-tree); t_i é o tempo, em milissegundos, somente de construção da Max-tree. Todos os testes de desempenho indicados foram feitos em uma mesma máquina[1] e os algoritmos, implementados em C/C++.

A Figura 6.7 exemplifica a detecção de linhas. A tolerância de espessura adotada foi: entre 1 e 5 para a imagem de fraturas no solo (primeira imagem negada); entre 1 e 5 para a imagem da régua (antes, foi feita abertura por área 50 sobre o negativo desta). A Figura 6.8 ilustra a detecção de retas. A máxima distância $d_{\max}$ dos pixels em $\mathcal{C}_{\mathcal{N}}$ até a reta suporte $\mathcal{L}$ foi de: 3 para a primeira imagem (e abertura por área 50); 4 para a imagem da calculadora; 1 para a imagem do carro (e abertura por área 30). A busca por círculos é mostrada na Figura 6.9, onde são estabelecidas tolerâncias de raios: entre 10 e 30 para a detecção de aspirinas; entre 3 e 7 para a bola de tênis de mesa; e entre 1 e 4 para a bola de futebol (e abertura por área menor que 5 e maior que 10). A Figura 6.10 consiste na detecção de arcos de raios: entre 100 e 115 para a imagem de um corredor; e entre 30 e 50 para a imagem da caixa de ferramentas. Finalmente, a Figure 6.11 exibe a detecção de elipses com tolerância de distância, sobre o lugar geométrico, de 10 para a primeira imagem e 3 para a segunda (ambas filtradas com abertura por área 500 inicialmente).

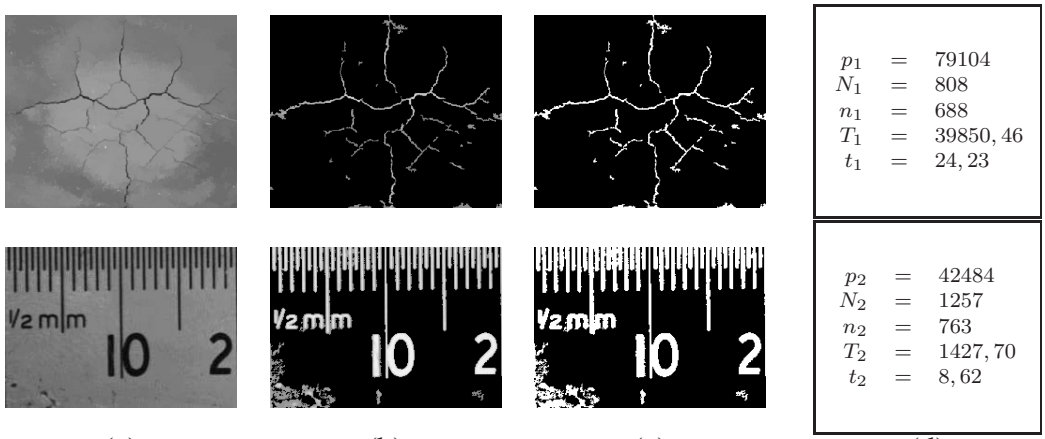
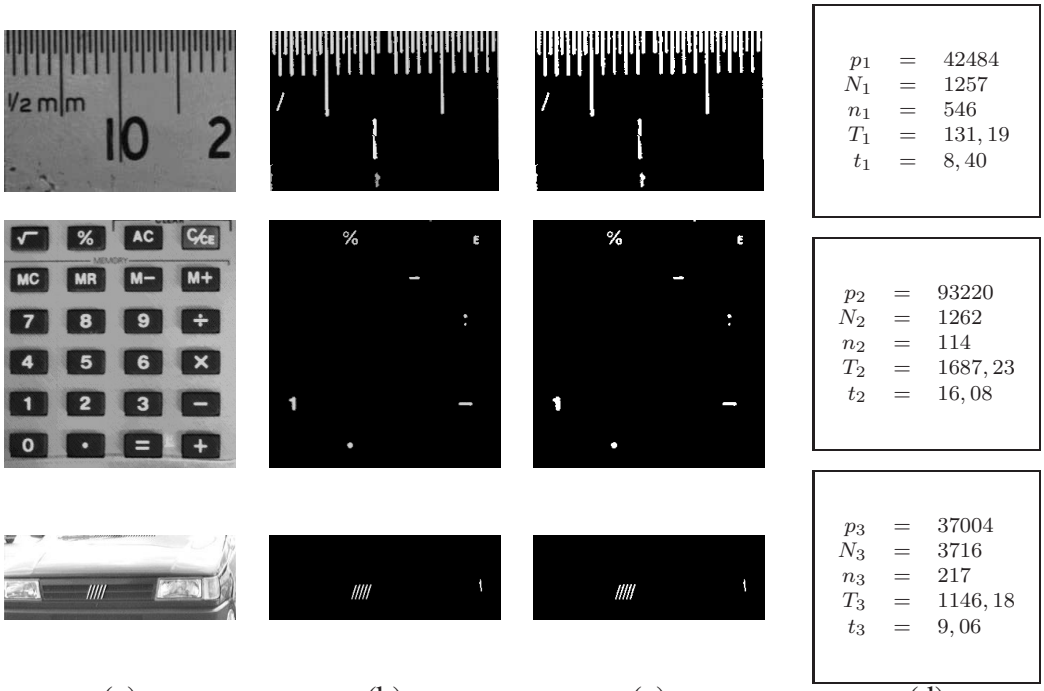


Fig. 6.7: Segmentação de linhas. (a) Imagem original I. (b) Imagem dos nós $\mathcal{N}_i$ selecionados de MT_I. (c) Imagem binária $\bigcup_{i=1}^{n} \mathcal{C}_{\mathcal{N}_i}$.

Estatísticas sobre as imagens processadas são apresentadas na Tabela 6.1. As colunas mostram: a relação média dos total de pixels e o total de nós (ou regiões) da imagem; a porcentagem média dos nós selecionados após a filtragem e detecção de forma; e o tempo médio, após a construção da Max-tree, destinado ao processo de reconhecimento de cada nó selecionado. Estes resultados reforçam o fato de que há menos elementos em relação à representação baseada em pixel (até $58,88$

[1]Pentium© Core™ 2 Duo, 2.0GHz, cache 2MB, RAM 3GB.



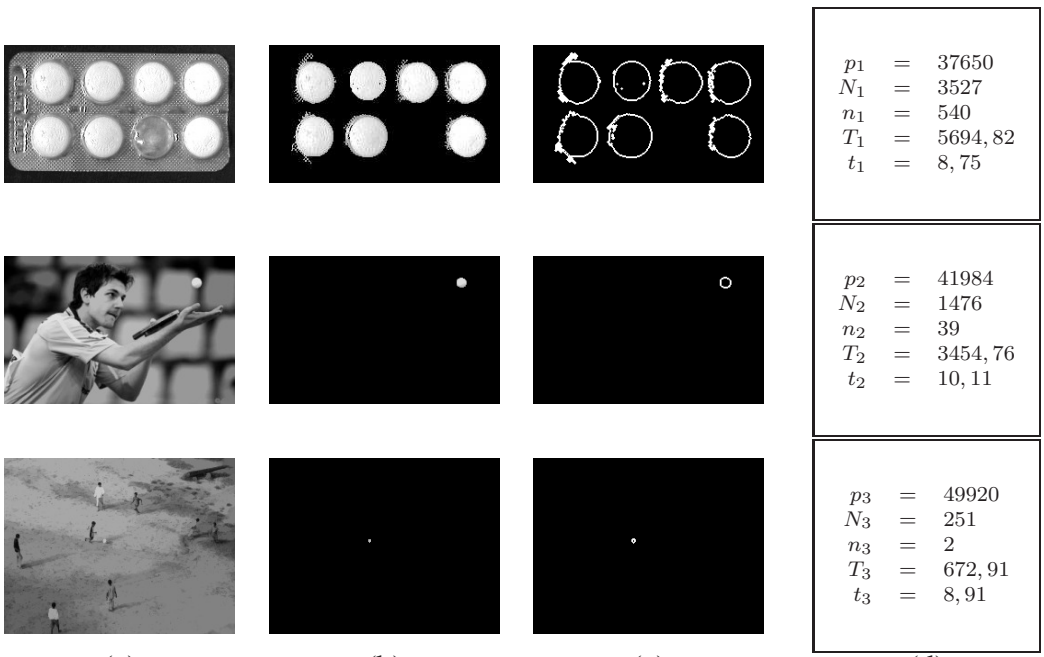
$p_1 = 42484$
$N_1 = 1257$
$n_1 = 546$
$T_1 = 131,19$
$t_1 = 8,40$

$p_2 = 93220$
$N_2 = 1262$
$n_2 = 114$
$T_2 = 1687,23$
$t_2 = 16,08$

$p_3 = 37004$
$N_3 = 3716$
$n_3 = 217$
$T_3 = 1146,18$
$t_3 = 9,06$

(a) (b) (c) (d)

Fig. 6.8: Segmentação de retas. (a) Imagem original I. (b) Imagem dos nós $\mathcal{N}_i$ selecionados de MT_I. (c) Imagem binária $\bigcup_{i=1}^{n} \mathcal{C}_{\mathcal{N}_i}$.



$p_1 = 37650$
$N_1 = 3527$
$n_1 = 540$
$T_1 = 5694,82$
$t_1 = 8,75$

$p_2 = 41984$
$N_2 = 1476$
$n_2 = 39$
$T_2 = 3454,76$
$t_2 = 10,11$

$p_3 = 49920$
$N_3 = 251$
$n_3 = 2$
$T_3 = 672,91$
$t_3 = 8,91$

(a) (b) (c) (d)

Fig. 6.9: Segmentação de círculos. (a) Imagem original I. (b) Imagem dos nós $\mathcal{N}_i$ selecionados de MT_I. (c) Contorno binário de todos os componentes conexos de interesse $\bigcup_{i=1}^{n} \Psi(\mathcal{C}_{\mathcal{N}_i})$.

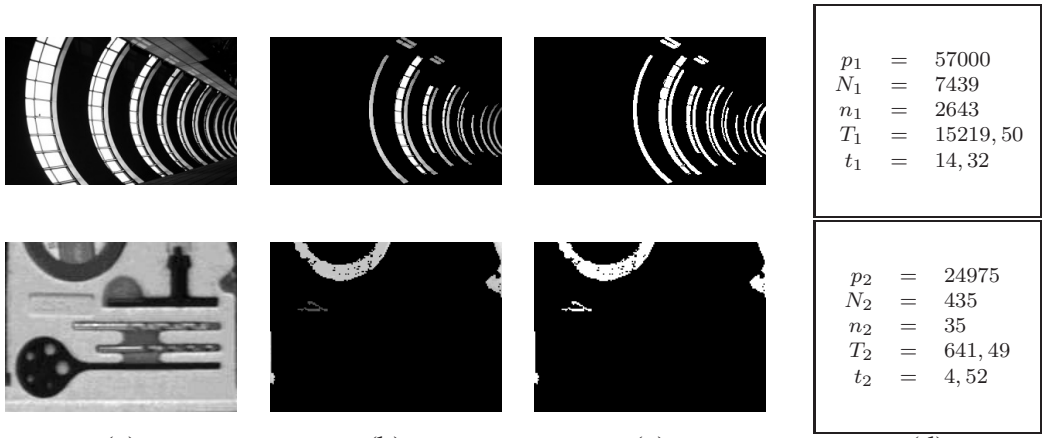


Fig. 6.10: Segmentação de arcos. (a) Imagem original I. (b) Imagem dos nós $\mathcal{N}_i$ selecionados de MT_I. (c) Imagem binária $\bigcup_{i=1}^{n} \mathcal{C}_{\mathcal{N}_i}$.

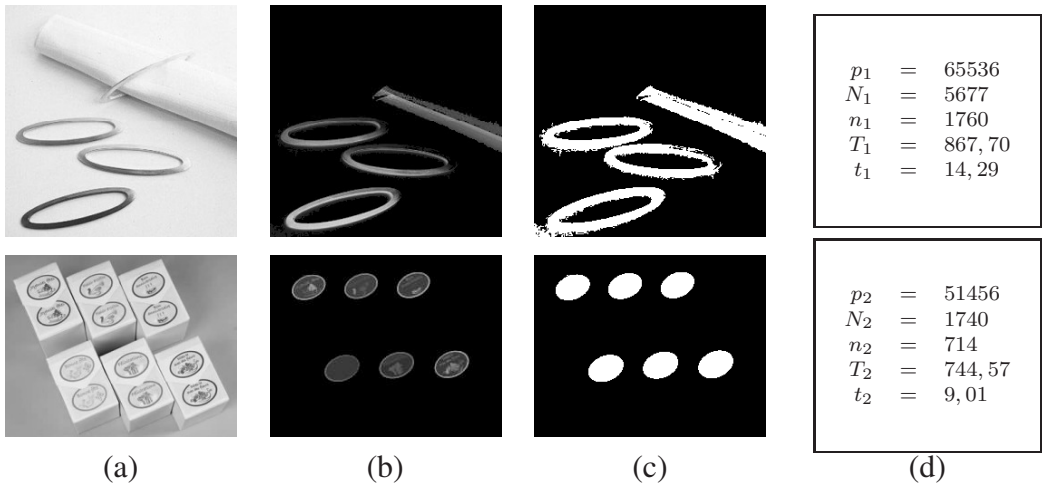


Fig. 6.11: Segmentação of elipses. (a) Imagem original I. (b) Imagem dos nós $\mathcal{N}_i$ selecionados de MT_I. (c) Imagem binária $\bigcup_{i=1}^{n} \mathcal{C}_{\mathcal{N}_i}$.

vezes menos no contexto dos experimentos). Além disto, uma grande quantidade de nós pode ser descartada no processo de filtragem e segmentação (por exemplo, somente $11,05\%$ de todos os nós são usados nos exemplos de detecção de círculos). Finalmente, o tempo médio de reconhecimento por nó selecionado indica que linhas e círculos necessitam de maior esforço computacional nos casos vistos. Mas isto depende da filtragem feita (quanto mais significativa, menos nós a processar) e da tolerância escolhida (quanto mais expressiva, mais provável de um nó próximo à raiz ser selecionado e, portanto, uma quantidade maior de descendentes não é tratada).

Tab. 6.1: Estatística das imagens em níveis de cinza processadas. Sequência das colunas: forma, pixel por nó, porcentagem de nós selecionados e tempo de reconhecimento por nó selecionado.

Forma	$\frac{\sum p_i}{\sum N_i}$	$100\frac{\sum n_i}{\sum N_i}$	$\frac{\sum T_i - \sum t_i}{\sum n_i}$
Linhas	$58,88$	$70,27\%$	$28,43$ms
Retas	$27,70$	$14,67\%$	$3,34$ms
Círculos	$24,66$	$11,05\%$	$20,36$ms
Arcos	$10,41$	$34,01\%$	$5,92$ms
Elipses	$15,77$	$33,36\%$	$0,64$ms

6.3 Considerações sobre o capítulo

A busca por formas de interesse pode ser útil em diversas aplicações: controle de qualidade em processos industriais, segmentação de fraturas geológicas, detecção de logomarca de veículos, identificação de marcações e objetos em imagens esportivas, reconhecimento de sinalizações para trajetória de robô, entre outros problemas de visão computacional. O algoritmo proposto é baseado na análise de regiões da Max-tree, onde a reconstrução rápida de componentes isolados e a exploração de hierarquia são considerados para um melhor desempenho. Filtragens descritas neste trabalho podem ser utilizadas para redução do número de nós e consequente simplificação da imagem, permitindo um menor tempo de processamento sem comprometer a extração da informação procurada. O tempo para construção da Max-tree, assim como as filtragens sobre a árvore, podem ser significativamente menores quando comparados ao tempo total de análise de forma para cada um dos componentes de nível.

A função para detecção da forma de um componente conexo pode ser ajustada para melhor precisão na segmentação. Algumas vantagens do método proposto sobre as técnicas tradicionais baseadas na transformada de Hough podem ser citadas: não há necessidade de prévia binarização da imagem (a transformada de Hough, por exemplo, é normalmente aplicada sobre a imagem binária resultante da segmentação de contornos); a complexidade não depende do número de parâmetros da

figura geométrica (não é necessária a criação de um espaço discreto de acumulação para os parâmetros envolvidos); formas não-parametrizáveis podem ser detectadas por análise morfológica dos componentes conexos ou casamento de modelos (a análise de forma não exige necessariamente que se leve em conta parâmetros utilizados na descrição de lugares geométricos detectáveis). Como desvantagens, pode-se citar: a limitação, pela técnica desenvolvida, de que as formas de interesse devam ter intensidades totalmente superiores (para Max-tree) ou inferiores (para Min-tree) às suas regiões vizinhas (por exemplo, um círculo dividido em duas metades, uma mais clara que o fundo e outra mais escura, apresenta componentes de nível apenas em forma de semicírculos); custo computacional elevado dependendo da análise de forma efetuada sobre a disposição de pixels (por exemplo, para a detecção de linhas, a transformada de distância é calculada para cada componente).

O problema de desempenho mencionado pode ser contornado com a divisão do conjunto total de nós da Max-tree em grupos, de maneira que cada grupo represente regiões da imagem com áreas similiares (tanto quanto possível), e se possa efetuar processamento paralelo de cada grupo, visto que a decisão sobre a existência de uma forma procurada é feita independentemente em cada nó [121]. Como trabalho futuro, pode-se pensar em uma divisão natural destes grupos a partir de subárvores isoladas conforme a segmentação da Max-tree baseada nas k maiores extinções por área, conforme o Algoritmo 5.2 (Cap. 5), supondo a disponibilidade de k processadores, um para cada ramo destacado da árvore.

Capítulo 7

Considerações finais

A Árvore de Componentes é uma importante ferramenta para representação, baseada em regiões, de imagens, possibilitando implementação eficiente de variados operadores conexos antiextensivos [1, 36]. A formalização da estrutura tratada neste trabalho data da segunda metade da década de 90. Desde então, diversos autores a utilizam, com destaque para operações de simplificação [7, 8, 9], filtragem por atributos [10, 15, 16], segmentação [5, 11, 12, 17, 18, 19] e reconhecimento de padrões [21, 29, 28, 30].

Apesar das várias aplicações funcionais mencionadas, uma nova implementação de construção da Max-tree (ou Árvore de Componentes) foi sugerida com o benefício de acrescentar eficientemente uma série de novos atributos e, ainda assim, ter desempenho e utilização de recursos de memória compatíveis com outras soluções. Isto sendo possível, em grande parte, em função das estruturas de dados utilizadas, envolvendo uso de *hashing* com base no histograma da imagem e alocação em bloco conforme a necessidade de novos nós na estrutura. Esta configuração, aliás, é importante no sentido de permitir uma previsibilidade de uso de recursos de memória mais fina (relação linear para o número de pixels vista nos gráficos do Cap. 3) que a de outras estruturas de dados em soluções anteriores na literatura.

Os atributos dos nós (componentes) são calculados de forma incremental, sem alterar a complexidade $O(N \times L)$ de construção da árvore, e servem de parâmetro, por exemplo, para o projeto de filtros. Estes são desenvolvidos no sentido de simplificar uma imagem de maneira eficiente, preservando estruturas principais (por exemplo, de nós onde ocorrem ramificações) ou mantendo o aspecto visual, segundo medidas objetivas de similaridade. As grandes vantagens desta abordagem são: **(i)** menos elementos a processar (visto que há significativamente menos nós do que pixels); **(ii)** renderização da imagem resultante após filtragens sucessivas ou paralelas (com dois ou mais atributos podendo ser observados para remoção de nós) no domínio da árvore. Operadores conexos antiextensivos foram apresentados considerando exclusivamente os atributos associados a cada nó, ou somente a topologia

das árvores, ou como uma combinação destes dois.

Em suma, tem-se, pela rica informação semântica introduzida em uma representação baseada em regiões da imagem, a possibilidade de: **(i)** implementação alternativa mais eficiente de algoritmos existentes (por exemplo, aninhamento de operações que podem ser feitas simultaneamente no domínio da árvore); **(ii)** desenvolvimento de novos algoritmos com amplos recursos de parametrização (por exemplo, simplificação da imagem, baseada na topografia ou estatística dos componentes e na topologia da árvore, com manutenção da similaridade ou aspecto visual).

Embora tenha se estabelecido variados operadores com amplas possibilidades de filtragem, com base em uma mesma estrutura, estes dificilmente se caracterizam como autossuficientes em aplicações mais elaboradas. Todos os operadores, sendo baseados em podas ou enxertos (vide Cap. 4) da árvore de componentes, têm as propriedades de antiextensividade e preservação de contornos em comum. Tentativas de aplicações completas, usando exclusivamente operadores desenvolvidos neste trabalho, para detecção de caracteres em placas de veículos ou do nariz em imagens *range* $2,5D$ de faces foram exaustivamente testadas, por terem configuração de relevo adaptada ao modelo. No entanto, invariavelmente detectava-se falhas em porção considerável de imagens (bases utilizadas com mais de uma centena de amostras), mesmo em casos mais simples nos quais as regiões de interesse estavam bem destacadas. Notou-se que, muitas vezes, era importante conectar ou desconectar componentes (com ajuda, por exemplo, de filtros não-lineares de máximo ou de mínimo) para contornar o problema, ações não contempladas no escopo do trabalho. De qualquer maneira, os operadores propostos se revelam ferramentas auxiliares valiosas essencialmente de pré-processamento, simplificação de imagens, marcação de máximos relevantes ou exploração de padrões.

Com o algoritmo para determinação de valores de extinção definido em função da Max-tree, pode-se generalizar a ideia desta medida de contraste de maneira simples, bastando fundamentalmente um atributo crescente presente em cada nó. Uma infinidade de atributos crescentes podem ser imaginados. Neste trabalho, foi dada atenção apenas àqueles que pudessem ser calculados incrementalmente em tempo de construção da árvore. Com isto, além das tradicionais extinções de altura – ou dinâmica [105, 2, 8] –, área e volume [106, 107], mais cinco foram propostas com interpretação topológica (altura de subárvore e número de descendentes de nó) ou geométrica (altura, largura e diagonal da caixa envolvente). Essas extinções têm aplicação importante de seleção de máximos (ou mínimos) relevantes como marcadores para evitar sobressegmentação da imagem. Além disto, propôs-se uma técnica de segmentação no domínio da árvore, baseada no particionamento desta em k ramos mais significativos, segundo a extinção adotada. Nas imagens tratadas (Cap. 5), este método foi suficiente para contagem ou separação de objetos de interesse de forma eficiente e com expressiva imunidade à ruído Gaussiano.

Por fim, padrões (formas geométricas, em especial) puderam ser detectados a partir de um algo-

ritmo genérico de observação de formas presentes em cada componente da árvore. O ganho obtido está basicamente na exploração da hierarquia para evitar processamento desnecessário de descendentes de nós selecionados. Com isto, métodos de filtragem, segmentação e reconhecimento de padrões puderam ser desenvolvidos sob uma metodologia unificada em torno da Max-tree.

7.1 Trabalhos futuros

Observou-se que alguns filtros são de ordem linear em relação ao número de nós. No entanto, um passo adicional de atualização de atributos deve ser levado em consideração. Porém, como este é feito de forma genérica (para todos os atributos) sem identificar, a princípio, os locais onde houve remoção de nós (toda a árvore é analisada), implica que todos os operadores passam a tempo proporcional a, no mínimo, $O(N \times L)$. Faz-se necessária, portanto, a atualização de atributos individualizada em cada filtro (por exemplo, se o mesmo alterou apenas um ramo da árvore, então somente este ramo e seus antecessores deveriam ser revisados).

Uma investigação mais criteriosa de novas aplicações para valores de extinção, associados a outras técnicas, deve ser feita. Pretende-se focar um problema efetivo, onde as habilidades da técnica de valores de extinção no que se refere ao apontamento direto de objetos de interesse (conforme suas composições estruturais fornecidas pela Max-tree) e à imunidade a ruídos sejam aspectos essenciais, em termos de robustez e desempenho, comparado com outras aproximações. A determinação automática de bons limiares de extinção pode ser desenvolvida para uma classe de problemas possivelmente por modelos de treinamento. O relacionamento topológico entre o nós, associado com estatísticas sobre o seus atributos, deve ser pesquisado para obtenção de ferramentas nesta direção.

Verificou-se também a inconveniência da exigência de contraste da forma que se queira detectar com o seu fundo na imagem. De outra maneira, formas visualmente existentes, acabam não sendo detectadas. Trabalhos futuros devem ser elaborados no sentido de analisar a possibilidade de combinação da Max-tree com a Min-tree para reduzir este problema. Ainda na linha de detecção de padrões, um algoritmo paralelo deve ser desenvolvido, baseado no particionamento da árvore pelas k maiores extinções, para promover ganho em eficiência de execução, visto que a porção de processamento da análise de forma efetuada sobre os componentes é expressivamente superior às demais etapas envolvidas (como, por exemplo, construção da Max-tree).

Por fim, seria interessante verificar a possibilidade de implementação eficiente de algumas operações morfológicas básicas, como casos particulares de dilatação e erosão, no domínio da árvore.

Referências Bibliográficas

[1] P. Salembier, A. Oliveras, and L. Garrido. Antiextensive Connected Operators for Image and Sequence Processing. *IEEE Transactions on Image Processing*, 7(4):555–570, 1998.

[2] F. Meyer. The dynamics of minima and contours. In *Mathematical Morphology and Its Applications to Image Processing*, pages 329–336. Kluwer Academic Publishers, 1996.

[3] M. Couprie and G. Bertrand. Topological grayscale watershed transformation. In *SPIE Vision Geometry V Proceedings*, volume 3168, pages 136–146, 1997.

[4] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for filtering, segmentation and information retrieval. In *IEEE Int. Conference on Image Processing*, Proc. of ICIP'98, pages 4–7, Chicago (IL), USA, October 1998.

[5] R. Jones. Connected filtering and segmentation using component trees. *Computer Vision and Image Understanding*, 75(3):215–228, 1999.

[6] J. Mattes and J. Demongeot. Efficient Algorithms to Implement the Confinement Tree. In 9^{th} *International Conference on Discrete Geometry for Computer Imagery*, pages 392–405, London, UK, 2000. Springer-Verlag.

[7] C. Vachier and L. Vincent. Valuation of image extrema using alternating filters by reconstruction. In *Neural, Morphological, and Stochastic Methods in Image and Signal Processing*, volume 2568 of *Proc. of SPIE*, pages 94–103, San Diego, CA, July 1995.

[8] G. Bertrand. On the dynamics. *Image and Vision Computing*, 25(4):447–454, 2007.

[9] A. G. Silva and R. A. Lotufo. New extinction values from efficient construction and analysis of extended attribute component tree. In *Proceedings of XXI SIBGRAPI*, pages 204–211, Campo Grande, Brazil, October 2008. IEEE Computer Society.

[10] E. J. Breen and R. Jones. Attribute Openings, Thinnings, and Granulometries. *Computer Vision and Image Understanding*, 64(3):377–389, November 1996.

[11] L. O. Garrido. *Hierarchical region based processing of images and video sequences: application to filtering, segmentation and information retrieval*. PhD thesis, Department of Signal Theory and Communications - Universitat Politècnica de Catalunya, Barcelona, Spain, April 2002.

[12] V. Mosorov and T. Kowalski. The development of component tree structure for grayscale image segmentation. In *Proceedings of the International Conference Modern Problems of Radio Engineering, Telecommunications and Computer Science*, pages 252–253, 2002.

[13] P. Dokladal, I. Bloch, M. Couprie, D. Ruijters, R. Urtasun, and L. Garnero. Topologically controlled segmentation of 3d magnetic resonance images of the head by using morphological operators. *Pattern Recognition*, 36(10):2463–2478, October 2003.

[14] M. A. G. Carvalho. *Análise Hierárquica de Imagens Através da Árvore dos Lagos Críticos*. Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação - Universidade Estadual de Campinas, Janeiro 2004.

[15] J. Darbon and C. B. Akgül. An efficient algorithm for attribute openings and closings. In *Proc. of the* 13^{th} *European Signal Processing Conference (EUSIPCO)*, Antalya, Turkey, September 2005. http://jerome.berbiqui.org/eusipco2005 (visitado em 06/11/09).

[16] D. Lesage, J. Darbon, and C.B. Akgul. An efficient algorithm for connected attribute thinnings and thickenings. pages 393–404, 2006.

[17] A. G. Silva, S. C. Felipussi, R. A. Lotufo, and G. L. F. Cassol. k-max: Segmentation based on selection of Max-tree deep nodes. In *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064 of *Proc. of IS&T/SPIE Electronic Imaging*, pages 195–202, San Jose, CA, January 2006.

[18] D. Menotti, L. Najman, and A. A. Araújo. 1d component tree in linear time and space and its application to gray-level image multithresholding. In 8^{th} *International Symposium on Mathematical Morphology (ISMM)*, volume 1, pages 437–448, São José dos Campos, October 10–13, 2007 2007. Universidade de São Paulo (USP), Instituto Nacional de Pesquisas Espaciais (INPE).

[19] B. Naegel, N. Passat, N. Boch, and M. Kocher. Segmentation using vector-attribute filters: methodology and application to dermatological imaging. In 8^{th} *International Symposium on Mathematical Morphology (ISMM)*, volume 1, pages 239–250, São José dos Campos, October 10–13, 2007 2007. Universidade de São Paulo (USP), Instituto Nacional de Pesquisas Espaciais (INPE).

[20] S. L. Kok-Wiles, M. Brady, and R. Highnam. Comparing mammogram pairs for the detection of lesions. In *Digital Mammography*, chapter 13. Kluwer Academic Publishers, 1998.

[21] J. Mattes, M. Richard, and J. Demongeot. Tree Representation for Image Matching and Object Recognition. In 8^{th} *International Conference on Discrete Geometry for Computer Imagery*, pages 298–312, Marne-la-Vallee, France, March 1999.

[22] M. Richard, M. Fleute, L. Desbat, S. Lavallée, and J. Demongeot. Registration of medical images for surgical action: use of optical sensors and matching algorithm. In J. J. Sanchez-Mondragon, editor, *Proc. SPIE, Sixth International Conference on Education and Training in Optics and Photonics*, volume 3831 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 125–139, June 2000.

[23] C. Berger, T. Geraud, R. Levillain, N. Widynski, A. Baillard, and E. Bertin. Effective component tree computation with application to pattern recognition in astronomical imaging. In *IEEE International Conference on Image Processing*, volume 4, pages 41–44, San Antonio, USA, September 2007.

[24] T. Meier and K. N. Ngan. Segmentation and tracking of moving objects for content-based video coding. *IEE Proceedings - Vision, Image, and Signal Processing*, 146(3):144–150, June 1999.

[25] L. Shi, Z. Zhang, and H. Wang. Automatic segmentation of video object plane based on object tracking and matching. In T. Zhang, B. Bhanu, and N. Shu, editors, *Proc. SPIE, Image Extraction, Segmentation, and Recognition*, volume 4550 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 28–33, September 2001.

[26] P. M. Roth, M. Donoser, and H. Bischof. On-line learning of unknown hand held objects via tracking. In *Proceedings of Second International Cognitive Vision Workshop*, 2006.

[27] P. Guillataud. *Contribution à l'analyse dendronique des images*. PhD thesis, Université de Bordeaux I, 1992.

[28] M. Léon, S. Mallo, and A. Gasull. A tree structured-based caption text detection approach. In *Proc. of the Fifth IASTED International Conference - Visualization, Imaging, and Image Processing*, Benidorm, Spain, September 2005.

[29] A. G. Silva and R. A. Lotufo. Detection of Lines Using Hierarchical Region Based Representation. In *Proceedings of XVII SIBGRAPI*, pages 58–64, Curitiba, Brazil, October 2004. IEEE Computer Society.

[30] A. G. Silva and R. A. Lotufo. Hierarchical Morphological Analysis for Generic Detection of Shapes in Grayscale Images. In *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, Proceedings of the International Symposium CompIMAGE 2006, pages 405–410, Portugal, 2006.

[31] A. Meijster, M. A. Westenberg, and M. H. F. Wilkinson. Interactive shape preserving filtering and visualization of volumetric data. In *Proc. of the Fourth IASTED Internation Conference - Signal and Image Processing*, Hawaii, USA, August 2002.

[32] G. K. Ouzounis and M. H. F. Wilkinson. Mask-Based Second-Generation Connectivity and Attribute Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):990–1004, 2007.

[33] M. A. Westenberg, J.B.T.M. Roerdink, and M.H.F. Wilkinson. Volumetric attribute filtering and interactive visualization using the max-tree representation. *IEEE Transactions on Image Processing*, 16(12):2943–2952, December 2007.

[34] M. Abdel-Mottaleb, N. Dimitrova, L. Agnihotri, S. Dagtas, S. Jeannin, S. Krishnamachari, T. McGee, and G. Vaithilingam. MPEG 7: A Content Description Standard Beyond Compression. In *IEEE Midwest Symposium on Circuits and System*, New Mexico, USA, August 1999.

[35] U. Braga-Neto and J. Goutsias. Grayscale Level Connectivity: Theory and Applications. *IEEE Transactions on Image Processing*, 13(12):1567–1580, 2004.

[36] L. Najman and M. Couprie. Building the Component Tree in Quasi-Linear Time. *IEEE Transactions on Image Processing*, 15(11):3531–3539, 2006.

[37] H. J. A. M. Heijmans. Introduction to Connected Operators. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Filters for Image Processing*, pages 207–235. SPIE–The International Society for Optical Engineering, 1999.

[38] E. R. Dougherty and R. A. Lotufo. *Hands-on Morphological Image Processing*. SPIE, 2003.

[39] A. C. Jalba, M. H. F. Wilkinson, and J. B.T.M. Roerdink. Morphological hat-transform scale spaces and their use in pattern classification. *Pattern Recognition*, 37:901–915, 2004.

[40] M. Wilkinson and J. Roerdink. Fast morphological attribute operations using tarjan's union-find algorithm. In *Mathematical Morphology and its Applications to Image and Signal Processing, Kluwer*, pages 311–320, 2000.

[41] A. Meijster and M. H. F. Wilkinson. A Comparison of Algorithms for Connected Set Openings and Closings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):484–494, 2002.

[42] O. Lezoray, C. Meurie, P. Belhomme, and A. Elmoataz. Multi-scale image segmentation in a hierarchy of partitions. In 14^{th} *EURASIP European Signal Processing Conference*, Florence, Italy, 2006.

[43] M. Carvalho, R. Lotufo, and M. Couprie. Spatiotemporal Segmentation of MR Image Sequence Based on Hierarchical Analysis. In *Seventh International Symposium on Signal Processing and Its Applications*, pages 677–680, Paris, France, July 2003.

[44] P. Salembier and J. Serra. Flat Zones Filtering, Connected Operators, and Filters by Reconstruction. *IEEE Transactions on Image Processing*, 4(8):1153–1160, August 1995.

[45] L. Vincent. Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, April 1993.

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

[47] R. Haralick, S. R. Sternberg, and X. Zhuang. Image Analysis Using Mathematical Morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, July 1987.

[48] M. Droogenbroeck and H. Talbot. Fast computation of morphological operations with arbitrary structuring elements. *Pattern Recognition Letters*, 17(14):1451–1460, December 1996.

[49] F. A. Zampirolli. *Transformada de Distância por Morfologia Matemática*. Tese de Doutorado, Faculdade de Engenharia Elétrica e de Computação - Universidade Estadual de Campinas, Junho 2003.

[50] L. di Stefano and A. Bulgarelli. A simple and efficient connected components labeling algorithm. *International Conference on Image Analysis and Processing*, 00:322–327, 1999.

[51] J. C. Hardwick. *Practical Parallel Divide-and-Conquer Algorithms*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1997. (visitado em 20/06/09).

[52] R. J. Sprugnoli. Perfect hashing functions: a single probe retrieving method for static sets. *Communications of the ACM*, 20:841–850, 1977.

[53] A. G. Silva, A. Fiorese, R. E. Silva, and G. B. Santos. Ane – Árvore n-ária de espalhamento naturalmente balanceada. *INFOCOMP Journal of Computer Science*, 6(2):81–90, June 2007.

[54] A. Klinger. Pattern and search statistics. *Optimizing Methods in Statistics*, pages 303–339, 1971.

[55] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, 1984.

[56] E. Shusterman and M. Feder. Image compression via improved quadtree decomposition algorithms. *IEEE Transactions on Image Processing*, 3(2):207–215, 1994.

[57] F. Meyer and S. Beucher. Morphological Segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, September 1990.

[58] L. Garrido, P. Salembier, and J. R. Casas. Representing and Retrieving Regions Using Binary Partition Trees. In *International Conference on Image Processing*, pages 605–609, Japan, October 1999.

[59] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576, 2000.

[60] J. Crespo, R. W. Schafer, J. Serra, C. Gratind, and F. Meyer. The flat zone approach: A general low-level region merging segmentation method. *Signal Processing*, 62(1):37–60, October 1997.

[61] L. Garrido, P. Salembier, and D. García. Extensive Operators in Partition Lattices for Image Sequence Analysis. *EURASIP Signal Processing*, 66(2):157–180, April 1998.

[62] B. Marcotegui. Segmentation Algorithm by Multicriteria Region Merging. In *Mathematical Morphology and Its Applications to Image Processing*, pages 313–320, Atlanta, USA, May 1996. Kluwer Academic Publishers.

[63] P. Salembier and F. Marquès. Region-based Representations of Image and Video: Segmentation Tools for Multimedia Services. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1147–1169, 1999.

[64] L. Chen, M. W. Berry, and W. W. Hargrove. Using Dendronal Signatures for Feature Extraction and Retrieval. *International Journal of Imaging Systems and Technology*, 11(4):243–253, 2000.

[65] C. Tzafestas and P. Maragos. Shape Connectivity: Multiscale Analysis and Application to Generalized Granulometries. *Journal of Mathematical Imaging and Vision*, 17:109–129, 2002.

[66] J. A. Bangham, P. D. Ling, and R. Harvey. Scale-Space From Nonlinear Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5), May 1996.

[67] J. R. Hidalgo. The representatin of images using scale trees. Master's thesis, University of East Anglia, Norwich, NR47TJ, UK, 1999.

[68] J. Bangham, K. Moravec, R. Harvey, and M. Fisher. Scale-space trees and applications as filters for stereo vision and image retrieval. In T. Pridmore and D. Elliman, editors, *British Machine Vision Conference*, pages 113–143, 1999.

[69] P. Monasse and F. Guichard. Fast Computation of a Contrast-Invariant Image Representation. *IEEE Transactions on Image Processing*, 9(5):860–872, 2000.

[70] P. F. Felzenszwalb and J. D. Schwartz. Hierarchical matching of deformable shapes. In *Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, June 2007. IEEE Computer Society.

[71] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

[72] P. M. Narendra and M. Goldberg. Image segmentation with directed trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:185–191, March 1980.

[73] A. X. Falcão, J. Stolfi, and R. A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):19–29, 2004.

[74] G. L. F. Cassol. Detecção de Formas Usando Representação Hierárquica de Regiões para Imagens em Níveis de Cinza. Trabalho de Conclusão de Curso, Universidade do Estado de Santa Catarina, Julho 2005.

[75] P. Salembier and L. Garrido. Connected Operators Based on Region-Tree Pruning Strategies. In 15^{th} *International Conference on Pattern Recognition*, volume 3, pages 371–374, Barcelona, Spain, September 2000.

[76] L. Garrido, A. Oliveras, and P. Salembier. Motion analysis of image sequences using connected operators. In *SPIE Visual Communications and Image Processing (VCIP)*, volume 3024, pages 546–557, San Jose, USA, February 1997.

[77] M. Donoser and H. Bischof. Efficient maximally stable extremal region (mser) tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 553–560. IEEE Computer Society, June 2006.

[78] S. J. F. Guimarães, M. Couprie, A. A. Araújo, and N. J. Leite. Video segmentation based on 2d image analysis. *Pattern Recognition Letters*, 24(7):947–957, 2003.

[79] S. J. F. Guimarães, M. Couprie, A. A. Araújo, and N. J. Leite. A method for cut detection based on visual rhythm. In *Proceedings of the 14th Brazilian Symposium on Computer Graphics and Image Processing*, pages 297–304, Florianópolis, Brazil, 2001. IEEE Computer Society.

[80] R. E. Tarjan. Efficiency of a Good But Not Linear Set Union Algorithm. *Journal of the ACM*, 22(2):215–225, 1975.

[81] W. H. Hesselink. Salembier's min-tree algorithm turned into breadth first search. *Inf. Process. Lett.*, 88(5):225–229, 2003.

[82] X. Huang, M. Fisher, and D. Smith. An Efficient Implementation of Max Tree with Linked List and Hash Table. In *VIIth Digital Imaging Computing: Techniques and Applications*, pages 299–308, Sydney, December 2003.

[83] SDC Information Systems. SDC Morphology Toolbox. 2008. http://www.mmorph.com (visitado em 06/11/09).

[84] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96, Columbia University, February 1996.

[85] K. Robinson and P. F. Whelan. Efficient morphological reconstruction: a downhill filter. *Pattern Recognition Letters*, 25(15):1759–1767, 2004.

[86] S. Yang, C. Wang, and X. Wang. Smoothing algorithm based on multi-scale morphological reconstruction for footprint image. In *Proceedings of the Second International Conference on Innovative Computing, Informatio and Control (ICICIC)*, page 525, Washington, DC, USA, 2007. IEEE Computer Society.

[87] E. R. Urbach, N. J. Boersma, and M. H. F. Wilkinson. Vector-attribute filters. volume 30 of *Computational Imaging and Vision*, pages 95–104. Springer-Verlag, Dordrecht, 2005.

[88] J. Crespo. *Morphological Connected Filters and Intra-Region Smoothing for Image Segmentation*. PhD thesis, Georgia Institute of Technology, November 1993.

[89] A. Meijster. *Efficient sequential and parallel algorithms for morphological image processing*. PhD thesis, Faculty of Mathematics and Natural Sciences - University of Groningen, Groningen, Netherlands, March 2004.

[90] C. Ronse and H. J. A. M. Heijmans. The Algebraic Basis of Mathematical Morphology – Part II: Openings and Closings. *Computer Vision, Graphics and Image Processing: Image Understanding*, 54:74–97, 1991.

[91] J. Serra and L. Vincent. An Overview of Morphological Filtering. *Circuits Systems Signal Process*, 11(1):47–108, 1992.

[92] L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In J. Serra and P. Salembier, editors, *Proc. EURASIP Workshop on Mathematical Morphology and Its Applications to Signal Processing*, pages 22–27. UCP Publications, 1993.

[93] J. Serra. Morphological filtering: an overview. *Signal Process*, 38(1):3–11, 1994.

[94] J. Crespo, J. Serra, and R. Schafer. Theoretical aspects of morphological filters by reconstruction. *Signal Processing*, 47(2):201–225, November 1995.

[95] A. Falcão, F. Bergo, and P. Miranda. Image segmentation by tree pruning. In *Proceedings of the XVII Brazilian Symposium on Computer Graphics and Image Processing*, pages 65–71, Washington, DC, USA, 2004. IEEE Computer Society.

[96] J. Crespo and V. Maojo. New Results on the Theory of Morphological Filters by Reconstruction. *Pattern Recognition*, 31(4):419–429, 1998.

[97] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[98] X. Huang, M. Fisher, and Y. Zhu. From min tree to watershed lake tree: Theory and implementation. In *International Conference on Image Analysis and Recognition*, pages 848–857, Porto, Portugal, September/October 2004.

[99] W. Niblack. *An Introduction to Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[100] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

[101] Q. Wu, F. Merchant, and K. Castleman. *Microscope Image Processing*. Elsevier/Academic Press, Amsterdam, 2008.

[102] S. C. Felipussi. *Modelagem de Sistemas Porosos por Meio da Geometria Estatística*. PhD thesis, Instituto de Informática - UFRGS, Porto Alegre, 2003.

[103] E. D. Ferrandière, S. Marshall, and J. Serra. Application of the morphological geodesic reconstruction to image sequence analysis. *IEE Proceedings: Vision, Image and Signal Processing*, 144(6):339–344, December 1997.

[104] S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. In E. R. Dougherty, editor, *Mathematical Morphology in Image Processing*, chapter 12, pages 433–481. Marcel Dekker, New York, 1993.

[105] M. Grimaud. A New Measure of Contrast: the Dynamics. In *Image Algebra and Morphological Image Processing III*, volume 1769, pages 292–305. SPIE–The International Society for Optical Engineering, 1992.

[106] C. Vachier and F. Meyer. Extinction value: a new measurement of persistence. In *IEEE Workshop on Nonlinear Signal and Image Processing*, volume I, pages 254–257, 1995.

[107] C. Vachier. *Extraction de caractéristiques, segmentation d'images et morphologie mathématique*. PhD thesis, École des Mines, Paris, France, 1995.

[108] A. G. Silva, R. A. Lotufo, and R. Arthur. Determinação eficiente de novos valores de extinção aplicados a problemas de visão computacional. In *XVII Congresso Brasileiro de Automática*, pages 1–6, Juiz de Fora, Brazil, October 2008.

[109] J. B. Mena. Automatic vectorization of segmented road networks by geometrical and topological analysis of high resolution binary images. *Know.-Based Syst.*, 19(8):704–718, 2006.

[110] J. Song, F. Su, and S. Cai. Raster to vector conversion of construction engineering drawings. *Automation in Construction*, 11(5):597–605, August 2002.

[111] Z. Sun, G. Bebis, and R. Miller. On-road vehicle detection using optical sensors: A review. In *IEEE International Conference on Intelligent Transportation Systems*, pages 125–137, 2004.

[112] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–766, 2008.

[113] M. Butzke, A. G. Silva, M. S. Hounssel, and M. A. Pillon. Automatic recognition of vehicle attributes color classification and logo segmentation. *Revista Hífen*, 32:293–300, 2008.

[114] L. A. P. Neves, J. M. de Carvalho, J. Facon, and F. Bortolozzi. Table-form extraction with artefact removal. *Journal of Universal Computer Science*, 14(2):252–265, 2008.

[115] P. Hough. In *Method and Means for Recognizing Complex Patterns*. U.S. patent 3069654, 1962.

[116] R. O. Duda and P. E. Hart. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[117] N. Aggarwal and W. C. Karl. Line Detection in Images Through Regularized Hough Transform. *IEEE Transactions on Image Processing*, 15(3):582–591, 2006.

[118] Hossam E. Abd El Munim and Aly A. Farag. A shape-based segmentation approach: An improved technique using level sets. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 930–935, Washington, DC, USA, 2005. IEEE Computer Society.

[119] A. G. Silva, R. A. Lotufo, and G. L. F. Cassol. Representação Hierárquica de Imagens para Detecção de Formas. In 11^{th} *Brazilian Symposium on Multimedia and the Web*, pages 223–225, Poços de Caldas, MG, 2005.

[120] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[121] A. H. S. Marcondes, M. A. Pillon, and A. G. Silva. Algoritmo de detecção de formas de interesse em imagens digitais para uma plataforma distribuída. *Revista Hífen*, 32:245–252, 2008.