Projeto de Operadores de Processamento e Análise de Imagens Baseados na Transformada Imagem-Floresta

Bruno Santos da Cunha

Dissertação de Mestrado

Projeto de Operadores de Processamento e Análise de Imagens Baseados na Transformada Imagem-Floresta

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Bruno Santos da Cunha e aprovada pela Banca Examinadora.

Campinas, 22 de Agosto de 2001.

Prof. Dr. Alexandre Xavier Falcão
IC - Universidade Estadual de Campinas
(Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Cunha, Bruno Santos da

C914p Projeto de operadores de processamento e análise de imagens baseados na transformada imagem-floresta / Bruno Santos da Cunha -- Campinas, [S.P. :s.n.], 2001.

Orientador : Alexandre Xavier Falcão

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de imagens. 2. Teoria dos grafos. 3. Morfologia (Matemática). I. Falcão, Alexandre Xavier. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Instituto de Computação
Universidade Estadual de Campinas

Projeto de Operadores de Processamento e Análise de Imagens Baseados na Transformada Imagem-Floresta

Bruno Santos da Cunha[1]

Agosto de 2001

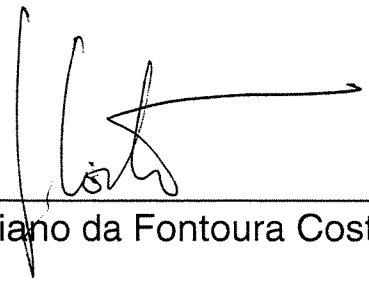
Banca Examinadora:

- Prof. Dr. Alexandre Xavier Falcão
  IC - Universidade Estadual de Campinas (Orientador)
- Prof. Dr. Luciano da Fontoura Costa
  IFSC - Universidade de São Paulo
- Prof. Dr. Luiz Marcos Garcia Gonçalves
  IC - Universidade Estadual de Campinas
- Prof. Dr. Neucimar Jerônimo Leite (Suplente)
  IC - Universidade Estadual de Campinas

[1]**FAPESP proc. 99/10100-3**

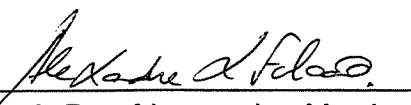
TERMO DE APROVAÇÃO

Tese defendida e aprovada em 06 de julho de 2001, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. Luciano da Fontoura Costa
IFSC - USP

Prof. Dr. Luiz Marcos Garcia Gonçalves
IC - UNICAMP



Prof. Dr. Alexandre Xavier Falcão
IC - UNICAMP

Prefácio

Diversos problemas em processamento e análise de imagens podem ser abordados como um problema de particionamento ótimo de uma imagem baseado em pixels sementes. A transformada imagem-floresta (IFT) se propõe a resolver tais problemas de maneira unificada e eficiente, a partir do cálculo de florestas de caminhos mínimos. Podendo ser considerada uma generalização de trabalhos voltados para a segmentação de imagens baseada em bordas, a IFT já foi utilizada para segmentação baseada em regiões, cálculo de linhas divisoras de águas e cálculo de transformadas de distância, inclusive baseadas na métrica Euclideana.

Neste trabalho acrescentamos novos operadores ao contexto da IFT, como métodos de segmentação de imagens baseada em conexidade fuzzy, geração de esqueletos multi-escala e operadores conexos. Realizamos comparações qualitativas e quantitativas com outros operadores descritos na literatura, além de apresentar exemplos de aplicações em imagens médicas e em vídeo digital. Exploramos também algumas questões de cunho teórico, como provas de corretude, análises de complexidades computacionais e garantias de qualidade do resultado de alguns métodos.

Abstract

In image processing and analysis, many problems can be thought of as an optimal image partition problem based on seed pixels, where each seed defines an influence zone composed by its "closest" pixels. The image foresting transform (IFT) is a unified and efficient approach to solve these problems, by reducing them into a shortest-path forest problem in a graph. It is an extension of previous works on boundary-based image segmentation methods, and it has already been used to design operators for region-based image segmentation, watershed transform and Euclidean distance transform.

In this work we add new image processing operators to the IFT framework, like image segmentation based on fuzzy connectedness, multiscale skeletons and connected operators. We make qualitative and quantitative comparisons with other operators described in the literature, and present some examples in medical imaging and digital video. We also explore some theoretical aspects, such as correctness proofs, complexity analysis and quality assurances of the results of some operators.

Agradecimentos

Gostaria de agradecer a todos que me apoiaram durante o desenvolvimento deste trabalho. À minha família e parentes, aos meus amigos, colegas e professores do Instituto de Computação da Unicamp. Não pretendo citar seus nomes para não correr o risco de me esquecer de algum deles enquanto escrevo este texto.

Mas agradeço em especial a meus grandes amigos Murilo e Inácia (pai e mãe), Anna Lúcia e Carolina (irmãs), Alexandre Falcão (orientador) e Cândida. Agradeço a eles pelo apoio nos momentos mais difíceis, às intermináveis conversas, aos preciosos conselhos. Não é exagero nenhum dizer que sem eles eu não teria chegado ao fim desta etapa. Muito obrigado.

Agradeço também à FAPESP por apoiar meu trabalho.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O uso de grafos [5, 6] para a resolução de problemas em processamento e análise de imagens e visão computacional tem sido investigado há vários anos. Apoiadas em uma sólida teoria e em uma grande quantidade de algoritmos eficientes, diversas soluções para vários destes problemas foram propostas.

Exemplos de sub-áreas onde grafos e/ou seus algoritmos têm sido utilizados são representação da topologia de formas [18], classificação por aglomeração (*clustering*) [28], segmentação de imagens [50, 63] e morfologia matemática [24, 65].

O problema de cálculo de caminhos de custo mínimo em grafos é um dos mais fundamentais em otimizações em redes. Algoritmos para a resolução deste problema têm sido estudados a bastante tempo [25, 26, 53], apesar de ainda ocorrerem avanços na teoria de algoritmos de caminhos mínimos [11]. No contexto de processamento e análise de imagens e visão computacional, estes algoritmos têm sido utilizados basicamente na perseguição de bordas [34, 54], caminhos geodésicos [43] e planejamento de trajetórias [14].

Diversos problemas podem ser abordados como um problema de particionamento ótimo de uma imagem a partir de pixels sementes. A transformada imagem-floresta (*IFT*, do inglês *image foresting transform*) [32] se propõe a resolver tais problemas. Podendo ser considerada uma generalização de trabalhos voltados para a segmentação de imagens baseada em perseguição de bordas [33, 34], a *IFT* já foi utilizada para a resolução de problemas como segmentação baseada em regiões [32], cálculo de linhas divisoras de águas (*LDA* ou *watershed*) [48] e transformadas de distância, inclusive baseadas na métrica Euclideana [49].

A *IFT* apresenta uma generalização do conceito de caminho de custo mínimo, e explora o cálculo de florestas de caminhos mínimos para a resolução de diversos problemas de uma maneira unificada e eficiente.

Neste trabalho acrescentamos novos operadores ao contexto da *IFT*, realizando comparações qualitativas e quantitativas com operadores descritos na literatura, além de

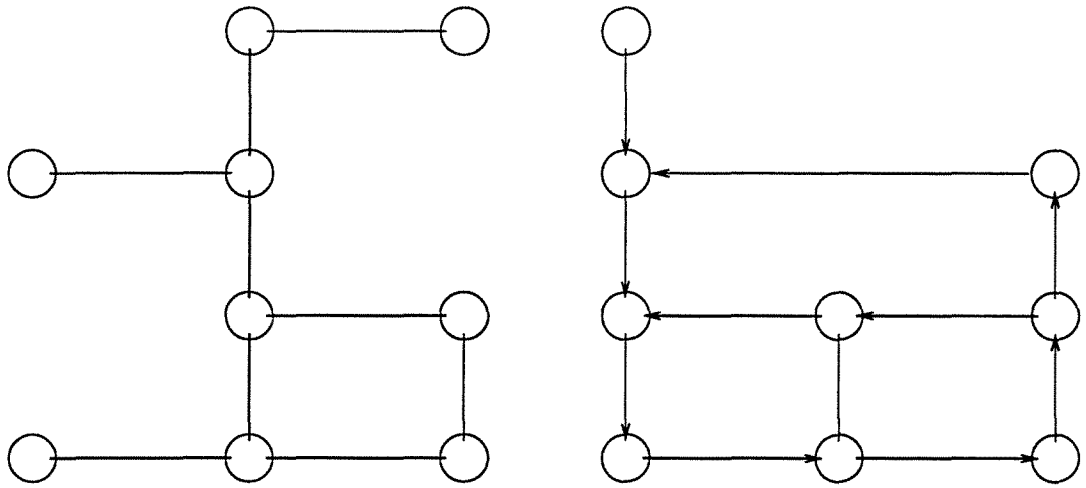


Figura 1.1: Exemplos de grafos. (a) grafo não-orientado; (b) grafo orientado.

apresentar exemplos de aplicações em imagens médicas e em vídeo digital. Exploramos também algumas questões teóricas como provas de corretude, análises de complexidade computacional e garantias de qualidade de alguns métodos.

A *IFT* está baseada em conceitos de teoria dos grafos, que não são muito comuns à área de processamento de imagens. Deste modo, vamos fazer uma rápida revisão de alguns conceitos que são necessários para a compreensão dos capítulos seguintes.

1.1 Alguns conceitos em teoria dos grafos

Os conceitos apresentados nesta seção seguem as definições apresentadas em [1]. Outras referências que podem levar a uma melhor compreensão destes conceitos são [6, 13].

Um grafo $G = (N, A)$ consiste em um conjunto N de nós (ou vértices) e um conjunto A de arcos (ou arestas) cujos elementos são pares de nós. Representamos a aresta que liga um nó p a outro nó q por (p, q). Se cada aresta é um par ordenado, dizemos que o grafo é orientado; caso contrário, que é não-orientado. Na Figura 1.1 temos um exemplo de grafo não-orientado e outro de grafo orientado.

A lista de adjacências $A(i)$ de um nó i é o conjunto de arestas que saem daquele nó, ou seja, $A(i) = \{(i, j) \in A : j \in N\}$. Se a aresta $(p, q) \in A$, dizemos que o nó q é adjacente ao nó p no grafo. Os arcos podem ter custos associados, sendo que neste caso o grafo também pode ser chamado de rede.

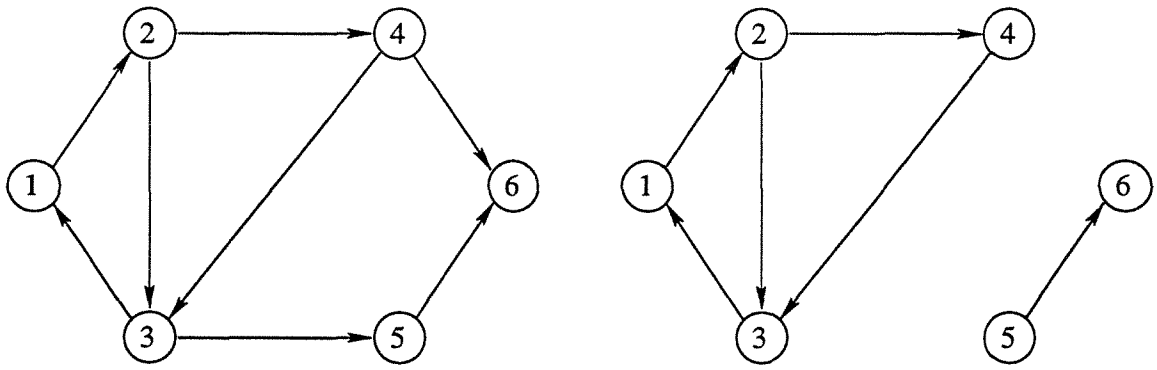


Figura 1.2: grafo conexo (a) e desconexo (b).

Um grafo $G' = (N', A')$ é um subgrafo de $G = (N, A)$ se $N' \subseteq N$ e $A' \subseteq A$. O grafo representado na Figura 1.2b é um subgrafo do grafo representado na Figura 1.2a.

Um caminho não orientado de v_1 a v_n no grafo é uma seqüência de nós $P = \langle v_1, v_2, \ldots, v_n \rangle$, sem repetições, onde, para $i = 1, \ldots, n-1$, ou a aresta $(v_i, v_{i+1}) \in A$ ou $(v_{i+1}, v_i) \in A$. Chamamos de caminho orientado de v_1 a v_n ao caminho não-orientado onde para cada $i = 1, \ldots, n-1$, a aresta $(v_i, v_{i+1}) \in A$. Neste trabalho quando falamos apenas caminho estamos nos referindo a um caminho orientado . O caminho $\langle 1, 2, 4, 6 \rangle$ na Figura 1.2a é orientado, enquanto que o caminho $\langle 3, 4, 6, 5 \rangle$ é não-orientado.

Dois nós i e j são conexos se o grafo contém pelo menos um caminho não orientado de i a j. Um grafo $G = (N, A)$ é conexo se todo par de nós $(p, q) \in N \times N$ é conexo. Caso contrário dizemos que ele é desconexo. Na Figura 1.2a temos um exemplo de grafo conexo, enquanto que o exemplo da Figura 1.2b é desconexo. Nos referimos aos subgrafos maximais conexos de um grafo como sendo suas componentes conexas. O grafo da Figura 1.2a possui uma componente conexa, enquanto que o grafo da Figura 1.2b possui duas.

Normalmente o custo de um caminho é dado pela soma dos custos dos arcos pertencentes ao caminho, correspondendo ao "comprimento" deste caminho. Neste trabalho, generalizamos este conceito através do uso de funções de custo de caminho. Uma função de custo de caminho $pf(P)$ é uma função não-negativa que associa um custo a um caminho P. Um caminho de custo mínimo - que chamaremos neste trabalho de caminho mínimo - entre dois nós i e j é aquele que minimiza a função de custo de caminho, dentre todos os possíveis caminhos de i a j em G. Definimos a distância entre i e j como sendo o custo do caminho mínimo entre estes nós, se tal caminho existir, ou ∞, caso contrário.

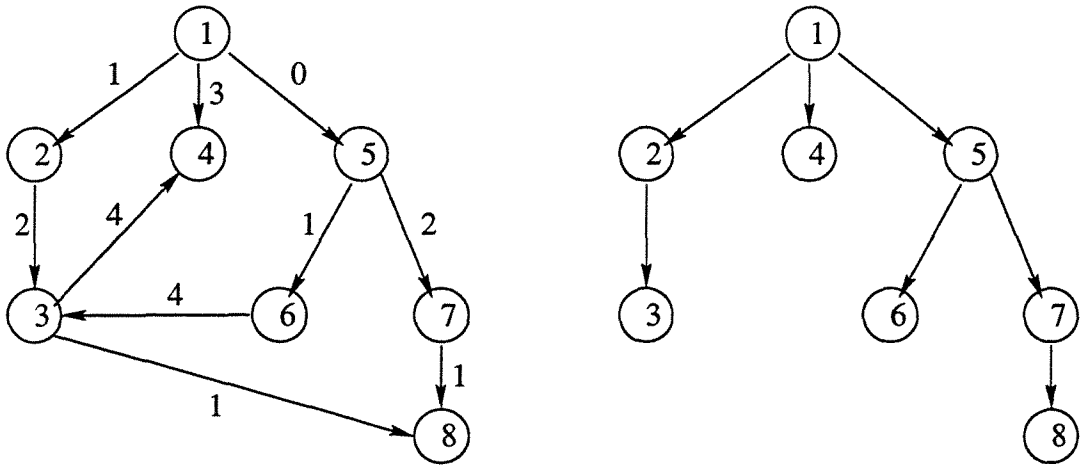


Figura 1.3: Árvore de caminhos mínimos enraizada no nó 1 (b) do grafo representado em (a).

Um ciclo orientado é formado por um caminho $\langle v_1, v_2, \ldots, v_n \rangle$ unido à aresta (v_n, v_1). Um grafo é acíclico se ele não contém nenhum ciclo orientado. A seqüência $\langle 1, 2, 3, 1 \rangle$ da Figura 1.2b é um ciclo orientado.

Chamamos um grafo acíclico e conexo de árvore. Definimos a raiz ou semente de uma árvore como sendo um vértice especial desta, que pode ser visto como o vértice a partir do qual a árvore "cresce". Uma árvore T é dita de caminhos de custo mínimo - ou simplesmente de caminhos mínimos - se ela é um subgrafo de um grafo G, onde o caminho (único) entre a raiz e qualquer vértice em T é um caminho de custo mínimo em G. Na Figura 1.3b temos representada uma árvore de caminhos mínimos do grafo representado na Figura 1.3a, a partir da semente (nó 1), utilizando como função de custo de caminho a soma dos pesos das arestas, com os pesos das arestas representados na Figura 1.3a.

Uma floresta é um grafo que não contém ciclos, sendo uma coleção de árvores. Assim, dizemos que uma floresta é de caminhos mínimos se suas componentes conexas são árvores de caminhos mínimos.

1.2 Estrutura do Trabalho

Inicialmente apresentamos a transformada imagem-floresta, mostrando que tipo de problema ela se propõe a resolver (capítulo 2). Como exemplo introdutório mostramos o

desenvolvimento de transformadas de distância através da IFT.

No capítulo 3, mostramos a utilização da IFT para a implementação de alguns dos métodos mais utilizados atualmente na área de segmentação de imagens, acrescentando à IFT os métodos baseados em conexidade *fuzzy*.

O desenvolvimento de um método para a geração de esqueletos multi-escala de alta qualidade é o tópico do capítulo 4.

Por fim, apresentamos no capítulo 5 as extensões da IFT para o desenvolvimento de operadores conexos, que permitem simplificar uma imagem sem criar bordas falsas, e, ao mesmo tempo, obter uma partição da imagem simplificada semelhante à que seria obtida se aplicássemos a transformada linhas divisoras de águas a partir de marcadores nesta imagem.

Incluímos ainda dois apêndices, relacionados à implementação dos métodos descritos. No apêndice A falamos sobre a estrutura de dados fila hierárquica, que permite que muitos dos métodos possuam complexidade de tempo linear no número de pixels da imagem. Já no apêndice B descrevemos duas interfaces gráficas, que permitem o cálculo de esqueletos multi-escala e utilização dos operadores conexos desenvolvidos.

Capítulo 2

A transformada imagem-floresta

A IFT é uma metodologia de resolução de problemas de processamento de imagens que consiste em reduzir o problema desejado ao cálculo de florestas de caminhos mínimos. Para tanto, a IFT trata uma imagem como se fosse um grafo, baseado em um modelo de grafo conveniente para o problema a ser resolvido. A partir de um conjunto de pixels sementes, uma floresta de caminhos de custo mínimo é calculada para este grafo, produzindo uma anotação da imagem. A idéia básica é construir operadores que resolvam facilmente o problema inicial utilizando esta anotação. As etapas deste processo estão representadas na Figura 2.1, e são discutidas nas seções 2.1 a 2.4. A seção 2.5 apresenta o algoritmo básico da transformada imagem-floresta, enquanto a seção 2.6 exemplifica aplicações da IFT para o cálculo de transformadas de distância, que serão necessárias no capítulo 4.

2.1 Modelando a imagem como um grafo e escolhendo parâmetros

A primeira etapa na resolução de um problema através da transformada imagem-floresta consiste na escolha de um modelo de grafo adequado ao problema. Para tanto, podemos construir explicitamente um grafo a partir de uma imagem e utilizar este grafo na IFT. No entanto, por questões de eficiência, preferimos apenas *pensar* na imagem como sendo um grafo, utilizando, portanto, uma representação implícita de grafos. A descrição que se segue supõe que a segunda representação seja utilizada. Esta representação se mostra adequada para problemas que trabalham diretamente com os pixels da imagem. Problemas de mais alto nível, como, por exemplo, os que utilizam regiões da imagem como elemento básico de processamento, podem precisar utilizar a representação explícita. Estes problemas fogem do escopo deste trabalho, e por isto não serão abordados.

Para tratarmos a imagem como um grafo, precisamos determinar quais elementos da

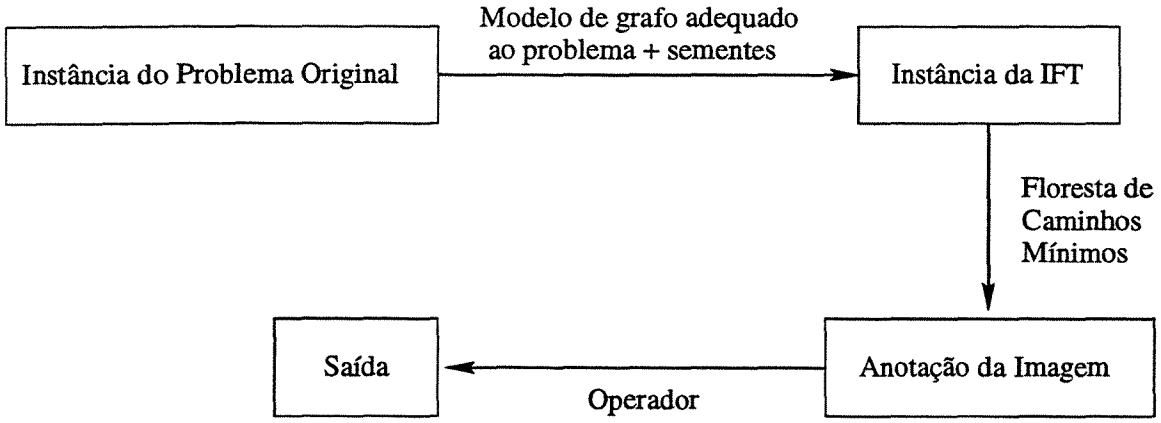


Figura 2.1: Esquema do funcionamento de operadores baseados na IFT: a idéia básica é que o problema inicial possa ser facilmente resolvido a partir da anotação da imagem.

imagem corresponderão aos vértices e às arestas do grafo, e de que forma o custo de se percorrer um caminho no grafo será calculado.

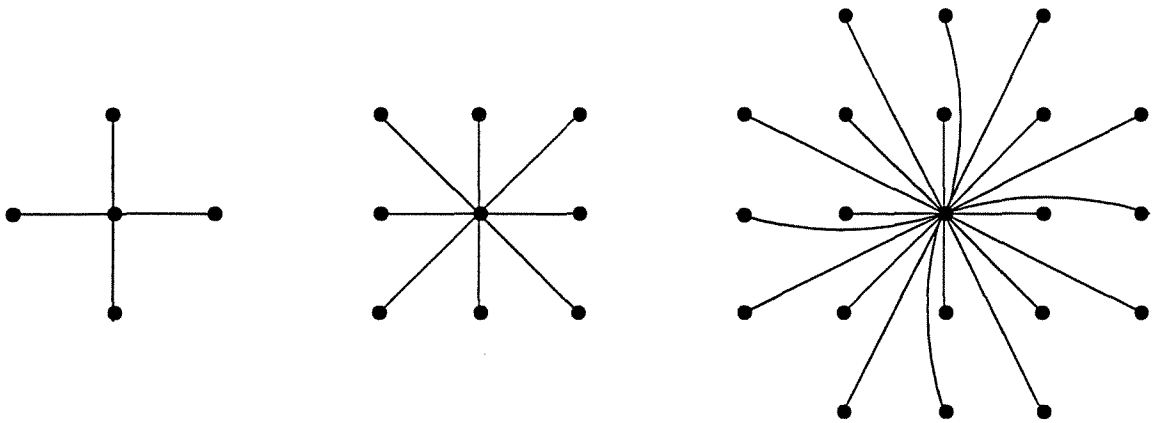
Consideramos que cada pixel da imagem corresponde a um vértice. Uma vez que estamos utilizando a representação implícita de grafos, chamamos indistintamente vértices de pixels e vice-versa. Já os arcos são definidos por uma *relação de adjacência* ρ entre pixels distintos. Podemos dizer, por exemplo, que dois pixels p e q são vizinhos se

$$d(p,q) \leq R, \tag{2.1}$$

onde $d(p,q)$ é a distância Euclideana entre p e q, e R é um número positivo. Se tomarmos $R = 1$, estaremos utilizando vizinhança 4 ; já se utilizarmos $R = \sqrt{2}$, vizinhança 8 [18, 35]. A Figura 2.2 ilustra alguns tipos de vizinhança possíveis. Para grande parte dos problemas considerados a utilização de vizinhança 4 ou 8 apresenta resultados satisfatórios. No entanto, é bom notar que em algumas aplicações pode ser interessante fazer com que todos os pixels sejam adjacentes entre si. Para tanto, basta escolhermos um valor de R adequado.

O uso da relação 2.1 determina que o grafo implícito possui topologia fixa para as arestas incidentes em cada vértice, a menos dos pixels próximos às bordas da imagem. Desta maneira, não é necessário que as arestas sejam armazenadas em alguma estrutura de dados, já que podemos calculá-las quando necessário.

Uma vez que iremos calcular florestas de caminhos de custo mínimo, utilizamos neste trabalho grafos ponderados, ou seja, grafos que possuem pesos nas arestas, representando o "comprimento" das mesmas. Assim, o custo de se percorrer uma aresta (p,q) (do nó p ao nó q) pode ser medido por um valor não-negativo $w(p,q)$. Como $w(p,q)$ pode ser



(a) (b) (c)

Figura 2.2: Algumas relações de adjacência obtidas a partir da Equação 2.1: $R = 1$ (vizinhança 4), $R = \sqrt{2}$ (vizinhança 8) e $R = \sqrt{5}$ (vizinhança 20).

diferente de $w(q,p)$, podemos modelar problemas que exijam grafos orientados. Os grafos considerados podem ter pesos fixos nas arestas, independentes dos caminhos tomados até se chegar a elas. No entanto, a IFT também pode, em algumas situações, atuar sobre grafos onde os pesos das arestas variam conforme o caminho escolhido da semente até um dado pixel. Um exemplo de grafo onde as arestas assumem pesos variáveis será apresentado na seção 2.6.

Associamos a cada caminho no grafo um custo de percorrê-lo, sendo calculado por uma *função de custo de caminho* $pf(P)$, que leva em consideração propriedades locais dos pixels do caminho, como os brilhos ou as posições destes pixels. Esta função pode ser escrita como função dos custos das arestas pertencentes ao caminho. A "distância" entre dois pixels é então dada como o "comprimento" do caminho de custo mínimo (ou simplesmente custo do caminho mínimo) entre o pixel origem e o pixel destino, sendo calculada através da minimização da função de custo de caminho. Exemplos de funções de custo de caminho são

$$pf(P) = \sum_{e \in P} w(e), \tag{2.2}$$

$$pf(P) = \prod_{e \in P} w(e), w(e) \geq 1, \tag{2.3}$$

e

$$pf(P) = \max_{e \in P} w(e), \tag{2.4}$$

onde P é o caminho considerado e $w(e)$ é o custo de se percorrer a aresta e.

A IFT requer ainda a escolha de um conjunto de pixels $S = \{s_1, s_2, \ldots, s_n\}$, que pode ser obtido de maneira automática ou através de interações com o usuário. Chamaremos cada elemento $s_i \in S$ de pixel semente, ou somente de semente. Exemplos de escolhas automáticas de sementes são os pixels pertencentes aos mínimos ou máximos regionais [65], no caso de imagens em nível de cinza, ou pixels pertencentes ao contorno dos objetos, no caso de imagens binárias (imagens nas quais os pixels assumem somente os valores 0 ou 1). Cada semente recebe um rótulo, que serve para identificar os pixels que estão em sua zona de influência. Duas ou mais sementes podem receber o mesmo rótulo, permitindo identificar zonas de influência associadas a objetos, quando estes objetos possuem mais de uma semente associada.

Operadores de processamento ou análise de imagens diferentes podem requerer modelos de grafos diferentes, isto é, escolhas diferentes de relações de adjacência, de sementes e de funções de custo de caminho.

Tendo visto quais são os parâmetros que devem ser definidos na escolha de um modelo de grafo adequado ao operador desejado, vamos apresentar qual é o problema que a IFT se propõe a resolver.

2.2 Particionando a imagem em árvores de caminhos mínimos

Considere que temos uma imagem de entrada I e um conjunto de sementes $S = \{s_1, s_2, \ldots, s_n\}$. Uma partição de um conjunto pode ser descrita como a lotação de cada elemento do conjunto em um subconjunto, de modo que a intersecção dos subconjuntos obtidos seja vazia e sua união resulte no conjunto original. Queremos particionar I em regiões R_i, de tal maneira que um pixel p pertença a R_i se a "distância" de s_i a p for menor que a "distância" de s_j a p, para todo j, com empates sendo resolvidos de acordo com alguma ordem de precedência assumida entre os pixels. Ou seja, o problema a ser resolvido consiste em se obter uma partição ótima de I a partir das sementes, de acordo com algum critério de "distância", com cada região obtida correspondendo à zona de influência da semente correspondente. Um exemplo de uma possível partição está representado na Figura 2.3, onde os pontos pretos representam as sementes e as zonas de influência são dadas em níveis de cinza diferentes.

A IFT aborda este problema como sendo uma partição de um grafo em árvores enraizadas nos pixels sementes, onde cada árvore é composta por uma semente e por todos

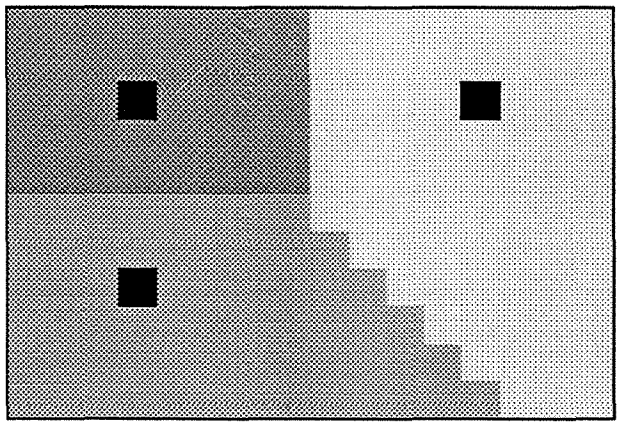


Figura 2.3: Exemplo de uma partição de uma imagem a partir de três sementes (representadas em preto). As zonas de influência associadas às sementes estão sombreadas.

os pixels que estão mais "próximos" àquela semente do que de qualquer outra – com empates sendo decididos de acordo com uma precedência assumida entre os pixels –, obtendo, assim, uma floresta de caminhos mínimos. Um exemplo de partição de grafo está representado na Figura 2.4. Na Figura 2.4a, temos um grafo com os pesos das arestas dados pelos valores próximos a elas. Aparecem ainda as sementes, representadas em preto. Neste exemplo, a função de custo de caminho utilizada é dada pela Equação 2.3. Na Figura 2.4b, temos representadas as zonas de influência de cada semente – dadas pelas árvores – e o custo do caminho de custo mínimo aparece próximo de cada vértice.

Uma maneira ingênua de se resolver este problema seria calcular a distância de cada semente a cada pixel da imagem, fazendo comparações para se determinar qual semente está mais próxima. Uma abordagem mais eficiente é propagar os valores de "distância" simultaneamente a partir de cada semente, até que zonas de influência de sementes distintas colidam, fazendo-se então a comparação somente para os pixels onde ocorreu a colisão. A segunda abordagem parece ainda mais eficiente quando lembramos que a "distância" entre uma semente e um pixel i pode não depender somente da posição da semente e de i na imagem (como no caso da distância entre dois pontos em um espaço Euclidiano), mas do custo do caminho mínimo entre a semente e i, de acordo com alguma função de custo de caminho. Desta maneira, enquanto a primeira abordagem equivale a se computar, a partir de cada semente, uma árvore de caminhos mínimos a todos os pixels da imagem, a segunda equivale a se computar uma floresta de caminhos mínimos na imagem, onde as árvores, enraizadas em pixels sementes, crescem simultaneamente, possuindo, portanto, custo computacional equivalente ao computo de apenas uma árvore de caminhos mínimos enraizada em uma semente e alcançando todos os pixels da imagem. Esta segunda abor-

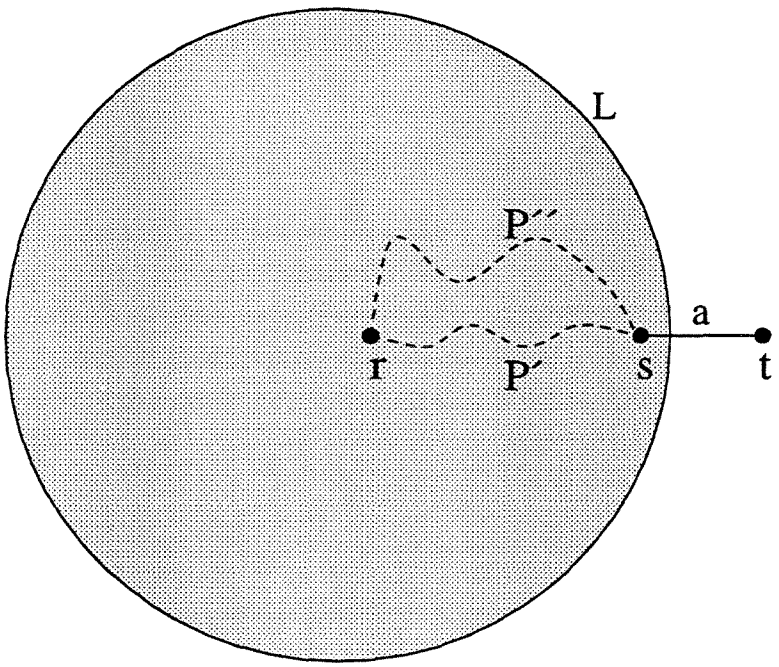


Figura 2.5: P' e P'' são caminhos de custo mínimo de r a s, e $P = P' \cdot \langle a \rangle$ é um caminho mínimo de r a t passando por s. O conjunto L contém todos os pixels que já terminaram de ser processados, em um dado instante.

pertencentes a L têm por propriedade o fato de que pelo menos um caminho de custo mínimo da raiz até l_i já foi encontrado, $1 \leq i \leq k$. Como os elementos que estão nesta lista já foram finalizados, eles não podem ser processados novamente. No caso de haver caminhos com custos decrescentes no grafo, poderia ser necessário que vértices pertencentes a L tivessem que ser analisados novamente, uma vez que poderiam ser alcançados por caminhos de custo menores do que os calculados anteriormente. Assim, a condição ($C1$), que diz que o valor da função de custo de caminho não deve ser decrescente ao longo do caminho, se torna clara.

A condição ($C2$) impõe que todo prefixo do caminho de custo mínimo calculado seja também um caminho de custo mínimo. Dito de outra forma, queremos que um caminho mínimo até um vértice t possa ser calculado a partir de uma extensão de um caminho de custo mínimo de r a s, com $s \in L$ no momento em que t é finalizado. Esta condição é necessária uma vez que a IFT calculará um caminho de custo mínimo de r a s, para qualquer vértice s antecessor de t em um caminho de custo mínimo de r a t.

Já a condição ($C3$) diz que o custo de um caminho mínimo não deve variar se escolhermos um outro prefixo, desde que o prefixo também seja um caminho de custo mínimo. Esta condição se dá do fato que dentre os (possíveis) diversos caminhos de custo mínimo de r a s, a IFT escolherá como caminho mínimo o primeiro que for encontrado. Deste modo, o caminho mínimo de r a t não deve depender de uma particular escolha de caminho mínimo de r a s.

Várias funções de interesse prático são suaves, como por exemplo a soma (dada na Equação 2.2)– quando os pesos das arestas são não negativos, como nos algoritmos de caminhos mínimos tradicionais [13, 1]–, a função produto (dada na Equação 2.3) – quando o peso das arestas é maior ou igual a um –, e a função aresta de peso máximo no caminho (Equação 2.4).

Assim, se a função de custo de caminhos for suave, a transformada imagem-floresta calculará uma floresta de caminhos mínimos, gerando uma anotação da imagem, que é utilizada na resolução do problema inicial.

2.4 A anotação da imagem

A IFT recebe os seguintes parâmetros: uma imagem de entrada, o conjunto de sementes S, uma função de custo de caminho pf e uma relação de adjacência, que estabelece os vizinhos de cada pixel. A saída é uma anotação da imagem, correspondendo à floresta de caminhos mínimos calculada. Esta floresta é representada através de três atributos para cada pixel p: um rótulo, que indica a qual região (ou árvore) o pixel pertence, o custo de um caminho mínimo da raiz de sua árvore até p, e um apontador para o pixel pai, que indica o antecessor de p no caminho de custo mínimo da semente mais próxima

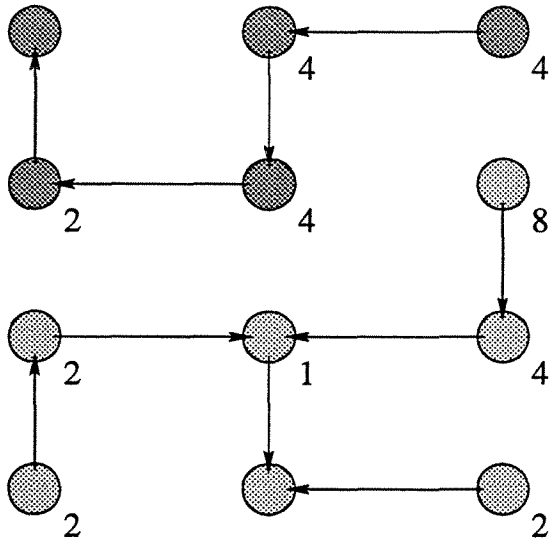


Figura 2.6: A anotação produzida para o exemplo da Figura 2.4: o rótulo de cada pixel está representado por tons de cinza, o custo do caminho de custo mínimo é dado ao lado de cada pixel e o apontador para o pixel pai é representado pela direção da seta. Saindo de um pixel e seguindo sempre na direção do pixel pai, terminamos por chegar à semente mais próxima àquele pixel.

até p. Atribuímos a cada semente um rótulo, e as informações presentes na anotação são computadas à medida que as árvores de caminhos mínimos crescem a partir das suas respectivas sementes. A anotação gerada para o exemplo da Figura 2.4 está representada na Figura 2.6.

Pelo menos um dos atributos da anotação da imagem deve ser relevante ao problema de processamento de imagens a ser resolvido. Por exemplo, problemas de segmentação baseados em perseguição de bordas utilizam o atributo pixel pai [33, 34]. Já o cálculo da transformada de distância Euclideana [49] e os operadores conexos [30] utilizam o custo do caminho mínimo. No cálculo da transformada de distância Euclideana, poderíamos também utilizar a informação de pixel pai para chegarmos ao pixel da borda do objeto mais próximo a um determinado pixel, seguindo o caminho que seria percorrido por uma gota de água que fosse solta naquela posição no mapa de distâncias. Problemas de segmentação baseada em regiões e o cálculo de esqueletos multi-escala [29] utilizam a informação de rótulo. Todos estes problemas são resolvidos pela IFT de maneira bastante eficiente.

Apresentaremos agora o algoritmo básico da transformada imagem-floresta, sua prova de corretude e uma análise de sua complexidade.

2.5 Algoritmo básico da IFT

Entrada: Uma imagem I, uma relação de adjacência ρ, um conjunto S de sementes rotuladas e uma função suave de custo de caminho $pf(\Pi_p)$, que calcula o custo de se percorrer o caminho Π_p de uma semente até o pixel p.

Saída: três imagens, *custo* (custos acumulados dos caminhos até cada pixel, em um dado instante), *pai* (pixel pai) e *rot* (rótulos), que representam a floresta computada.

Estruturas de dados auxiliares: uma fila de prioridades Q, uma lista L de pixels que já terminaram de ser processados e uma variável auxiliar *tmp*.

início

1. para todo pixel $p \in I$, atribua ∞ para $custo[p]$ e *nil* para $rot[p]$ e $pai[p]$;
2. para todo pixel semente $s \in S$, atribua 0 para $custo(s)$, o rótulo associado a s para $rot[s]$, e insira s em Q;
3. *enquanto* Q não estiver vazia *faça*
   a. remova o pixel p de Q tal que $custo[p] = \min_{p' \in Q}\{custo[p']\}$ e insira p em L;
   b. *para* cada pixel q adjacente a p de acordo com ρ, $q \notin L$, *faça*
      (i) calcule $tmp \leftarrow pf(\Pi_p \cdot \langle (p,q) \rangle)$, representando o custo do caminho para se chegar a q passando por p;
      (ii) *se* $tmp < custo[q]$ *então*
         a. atribua *tmp* para $custo[q]$, $rot[p]$ para $rot[q]$ e p para $pai[q]$;
         b. *se* $q \notin Q$ *então* insira q em Q *senão* atualize a posição de q em Q;
         fim se;
      fim para;

   fim enquanto;

fim

Prova 2.5.1 Corretude do algoritmo

Podemos pensar, a princípio, que qualquer função não decrescente de custo de caminho pode ser minimizada pelo algoritmo da IFT. Para mostrar que isto não é verdade, vamos

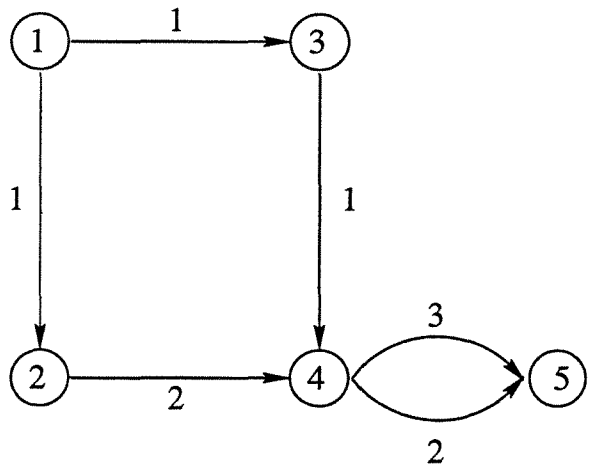


Figura 2.7: Contra-exemplo para a suposição de que toda função não decrescente de custo de caminho pode ser minimizada pelo algoritmo da IFT.

recorrer ao contra-exemplo da Figura 2.7. Suponha que queiramos calcular os caminhos mínimos a partir do nó 1 a todos os outros nós. Suponha ainda que a função de custo de caminhos é a soma do peso dos arcos, mas que estes pesos dependam do prefixo tomado até uma dada aresta. Vamos considerar que a aresta $(4,5)$ tem custo 3 se o caminho escolhido até o nó 4 for $\langle 1,3,4\rangle$ e 2 se o caminho for $\langle 1,2,4\rangle$. Desta forma, o algoritmo da IFT escolheria como caminho mínimo de 1 a 5 o caminho $\langle 1,3,4,5\rangle$, com custo 6, enquanto que haveria no grafo um caminho com custo 5, $\langle 1,2,4,5\rangle$. Este problema ocorre porque o caminho mínimo de 1 a 5 tem como prefixo um caminho que não é mínimo, $\langle 1,2,4\rangle$. Assim, fica claro que devemos impor mais restrições nas funções de custo de caminho.

Seja S o conjunto de pixels sementes. Para tornar a notação mais precisa, vamos denotar um caminho (não necessariamente de custo mínimo) de um vértice pertencente a S até um vértice x por Π_x.

O algoritmo da IFT satisfaz os seguintes invariantes, imediatamente antes da execução do passo 3a:

(I1) Para cada pixel $p \in L$, Π_p é um caminho de custo mínimo de S a p no grafo, codificado na imagem *pai*.

(I2) A fila Q contém todos os pixels $p' \notin L$ alcançáveis a partir de S por caminhos cujos vértices intermediários estão todos em L.

Estes invariantes são provados por indução no número de pixels $p \in L$, que é zero no inicio e é aumentado em uma unidade a cada iteração do passo 3. A base da indução é a

iteração inicial, onde L está vazia e somente s está em Q, sendo os dois invariantes trivialmente satisfeitos. Para o passo de indução, devemos mostrar que I1 e I2 são preservados pelos passos 3a e 3b.

Para provar uma proposição do tipo

$$a \Rightarrow b, \tag{2.5}$$

podemos provar a proposição equivalente

$$\neg b \Rightarrow \neg a. \tag{2.6}$$

Chamamos uma proposição do tipo dado pela equação 2.6 de contrapositiva da proposição dada na equação 2.5. Para provar que o algoritmo da IFT computa o resultado correto sempre que pf for uma função suave de custo de caminho, vamos provar a afirmação contrapositiva: *se o algoritmo da IFT está incorreto, então pf não é uma função suave de custo de caminhos.*

Seja q o próximo pixel a ser removido de Q no passo 3a, com $\Pi_q = \Pi_p \cdot \langle p, q \rangle$, $p \in L$ e Π_p caminho mínimo de S a p. O invariante I1 pode ficar falso pelos passos 3a e 3b se existir um pixel p' tal que $pf(\Pi_q) = pf(\Pi_{p'} \cdot \langle p', q \rangle) < pf(\Pi_q)$, significando que Π_q não é um caminho de custo mínimo de S a q. Vamos admitir que tal pixel p' existe, e avaliar as seguintes possibilidades:

(P1) Se $\Pi_{p'}$ não é um caminho de custo mínimo até p', então a condição C2 é violada (ver seção 2.3). Assim, $\Pi_{p'}$ deve ser um caminho de custo mínimo.

(P2) Se $\Pi_{p'}$ é um caminho de custo mínimo e $p' \notin L$, então a condição C1 é violada: uma vez que q é o próximo pixel a ser removido, $pf(\Pi_q) \leq pf(\Pi_{p'})$; no entanto, $pf(\Pi_{p'} \cdot \langle p', q \rangle) < pf(\Pi_q)$ implica $pf(\Pi_{p'} \cdot \langle p', q \rangle) < pf(\Pi_{p'})$ violando a condição C1. Assim, p' deve pertencer a L.

(P3) Se $\Pi_{p'}$ é um caminho de custo mínimo e $p' \in L$, então p' deve ser igual a p, pois caso contrário $pf(\Pi_{p'} \cdot \langle p', q \rangle) < pf(\Pi_p \cdot \langle p, q \rangle)$ implicaria que p não precederia q, por causa do passo 3biia do algoritmo. No entanto, $p' = p$ significa que existe um caminho $\Pi'_p \neq \Pi_p$ tal que $pf(\Pi'_p \cdot \langle p, q \rangle) < pf(\Pi_p \cdot \langle p, q \rangle)$, violando a condição C3.

Assim, se p' existe (e neste caso o algoritmo da IFT não está correto), então pf não é uma função suave de custo de caminhos. Deste modo, se pf é suave, Π_q é um caminho de custo mínimo de S a q no grafo, validando o invariante I1. A inserção de q em L no final do passo 3a não destrói o invariante I2. Uma vez que o passo 3b preserva $p \in L$ e os caminhos de custo mínimo Π_p para todo $p \in L$, concluímos que I2 também será valido na próxima iteração, o que completa a prova.

Análise 2.5.1 Complexidade Computacional

Vamos agora fazer uma análise das complexidades de tempo e espaço do algoritmo. Fica claro que o gargalo do algoritmo está na manutenção da fila de prioridades Q. No caso geral, pode ser interessante implementá-la como sendo um *heap* binário[13]. Se n é o número de vértices (ou seja, de pixels) e m é o número de arestas, o algoritmo levará tempo $O(n \log n + m \log n)$, pois são feitas n remoções da fila, cada uma levando tempo $O(\log n)$, e cada uma das m arestas direcionadas é visitada uma vez, podendo causar uma inserção ou atualização na posição do vértice na fila de prioridades, com cada operação tendo um custo $O(\log n)$, levando, portanto, ao tempo total mencionado.

No entanto, quando a função de custo de caminho assume apenas valores inteiros e a diferença entre o valor desta função entre quaisquer dois vértices consecutivos de um caminho é limitada a um inteiro K, a fila Q pode ser implementada como uma fila hierárquica de prioridades, como na implementação de Dial [1, 33] para o algoritmo de Dijkstra. Neste caso, o algoritmo tem complexidade de tempo $O(m + nK)$. Tal complexidade se deve do seguinte fato: cada aresta (direcionada) é verificada apenas uma vez, gerando uma complexidade $O(m)$, pois cada inserção ou atualização da posição do vértice na fila leva tempo $O(1)$. Além do mais, cada vértice sai da fila uma única vez, mas podemos ter que olhar em até $K-1$ posições (*buckets*) na fila até encontrarmos o vértice que deve sair da fila (aquele de menor custo acumulado), podendo ser necessárias, portanto $K-1$ vezes $O(n) = O(nK)$ operações, chegando à complexidade anunciada. Se utilizarmos uma relação de adjacência como a dada na relação 2.1, com R independente das dimensões da imagem, temos que o número de arestas é $O(n)$. Neste caso, temos que, sendo K uma constante (independente da entrada), a IFT leva tempo $\Theta(n)$ para calcular a floresta de caminhos mínimos, que é um tempo ótimo para a resolução da IFT uma vez que somente para gerar as imagens *custo*, *rot* e *pai* já é necessário um tempo de processamento $\Omega(n)$.

A memória necessária para o algoritmo é $O(n)$, uma vez que são geradas apenas três imagens do tamanho da imagem de entrada (que possui n pixels) e que tanto a fila hierárquica de prioridades quanto o *heap* ocupam espaço $O(n)$.

Falamos anteriormente que quando estamos calculando a partição, empates entre as distâncias de duas ou mais sementes até um determinado pixel são resolvidos de acordo com uma ordem de precedência assumida entre as sementes. A utilização de um critério de precedência permite que a partição calculada seja sempre a mesma. Para implementarmos este critério, ordenamos as sementes de acordo com a relação de precedência antes de chamar a IFT , e fazemos com que os pixels que possuem um mesmo valor de função de custo de caminho saiam da fila na mesma ordem em que entraram, seguindo uma abordagem FIFO (do inglês *first-in first-out*).

Devido à importância da utilização da fila hierárquica de prioridades para uma implementação eficiente da IFT, descrevemos melhor esta estrutura no Apêndice A.

Apresentamos agora a aplicação da transformada imagem-floresta ao problema do cálculo de transformadas de distância, uma vez que a transformada de distância Euclideana está na base do cálculo de esqueletos multi-escala a ser apresentado no capítulo 4.

2.6 Transformadas de Distância

Uma transformada de distância é uma operação que associa a cada pixel em uma imagem binária a distância deste pixel ao pixel mais próximo pertencente a uma borda do objeto, de acordo com alguma métrica. A imagem formada por estas distâncias é chamada de mapa de distâncias. Tais transformadas são úteis a diversas operações em processamento e análise de imagens, tais como cálculo de esqueletos baseados em mapas de distância, interpolação baseada em forma, dilatações, erosões e operações de classificação.

Apesar da métrica Euclideana ser a mais intuitiva, os algoritmos que a calculavam se mostraram, até recentemente, muito complexos ou ineficientes, levando à utilização mais freqüente de métricas mais simples, como as distâncias *city-block*, *chessboard* e chamfer.

Descreveremos nesta seção a implementação de transformadas de distância através da IFT que utilizam estas métricas. A revisão apresentada nesta seção não pretende ser completa, mas mostrar apenas que a transformada imagem-floresta pode calcular com grande facilidade e de maneira eficiente a transformada de distância para diversas métricas. Para uma boa revisão de transformadas de distância, sugerimos consultar [14].

2.6.1 Métrica *city-block*

A distância D_4 (*city-block*) entre dois pixels $p = (x_p, y_p)$ e $q = (x_q, y_q)$ é definida como:

$$D_4(p,q) = |x_p - x_q| + |y_p - y_q|. \tag{2.7}$$

Nesta métrica, os pixels com uma distância menor ou igual a um valor r a um pixel $p = (x_p, y_p)$ formam um diamante ao redor do pixel p, como o exemplo apresentado na Figura 2.8, onde estão representados os pixels com distância D_4 menor ou igual a 2.

Para modelarmos a IFT para calcular esta transformada de distância, basta utilizarmos a relação de adjacência vizinhança 4 (relação 2.1, com $R = 1$), w(p,q) = 1 e função custo de caminho soma (Equação 2.2).

2.6.2 Métrica *chessboard*

A distância D_8 (*chessboard*) entre dois pixels $p = (x_p, y_p)$ e $q = (x_q, y_q)$ é definida como:

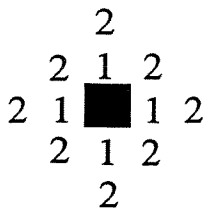


Figura 2.8: Pixels com uma distância *city-block* ≤ 2 ao pixel representado em preto. Estes pixels apresentam uma forma de diamante.

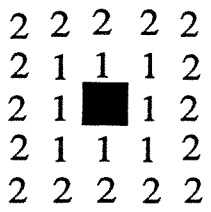


Figura 2.9: Pixels com uma distância *chessboard* ≤ 2 ao pixel representado em preto. Estes pixels apresentam uma forma de quadrado.

$$D_8(p,q) = \max(|x_p - x_q|, |y_p - y_q|). \tag{2.8}$$

Nesta métrica, os pixels com uma distância menor ou igual a um valor r a um pixel $p = (x_p, y_p)$ formam um quadrado ao redor do pixel p, como o exemplo apresentado na Figura 2.9, onde estão representados os pixels com distância D_8 menor ou igual a 2.

Para modelarmos a IFT para calcular esta transformada de distância, basta utilizarmos a relação de adjacência vizinhança 8 (relação 2.1, com $R = 1$), w(p,q) = 1 e função custo de caminho soma (Equação 2.2).

2.6.3 Métrica de *chamfer*

A distância $D_{cha(A:B)}$ (*chamfer*) entre dois pixels $p = (x_p, y_p)$ e $q = (x_q, y_q)$, utilizando-se vizinhança 8, é definida como:

$$D_{cha(A:B)}(p,q) = A \max(|x_p - x_q|, |y_p - y_q|) + (B - A) \min(|x_p - x_q|, |y_p - y_q|). \tag{2.9}$$

A distância de chamfer tenta atribuir aos pixels distâncias relativas mais próximas às que seriam atribuídas pela métrica Euclideana. As constantes A e B podem ser escolhidas

de maneira a garantir que a diferença entre a distância Euclideana e a distância de chamfer possuam um determinado limite superior. Segundo Cuisenaire [14], Borgefors sugere que utilizemos o valor 3 para A e 4 para B, uma vez que apenas valores inteiros são utilizados e que temos a garantia de que a distância de chamfer com estes valores excede os valores obtidos pela distância Euclideana em no máximo $0.08M$, com $M = \max(|x_p - x_q|, |y_p - y_q|)$.

Para modelarmos a IFT para calcular esta transformada de distância, basta utilizarmos a relação de adjacência vizinhança 8 (relação 2.1, com $R = 1$), função custo de caminho soma (Equação 2.2), e uma atribuição de peso às arestas que penalize mais as arestas diagonais do que as arestas horizontais ou verticais. Para $D_{cha(3:4)}$, devemos ter $w(p,q) = 3$ se p e q são vizinhos-4 e $w(p,q) = 4$ se p e q são vizinhos-8 mas não vizinhos-4. Vale notar que as distâncias *city-block* e *chessboard* são casos especiais da distância de chamfer ($D_{cha(1:2)}$ e $D_{cha(1:1)}$, respectivamente).

2.6.4 Métrica Euclideana

As transformadas de distância utilizando as métricas já apresentadas são bastante conhecidas, existindo a algum tempo algoritmos bastante eficientes para calculá-las. No caso da métrica Euclideana, somente a pouco tempo é que apareceram algoritmos simples e eficientes para o seu computo [15, 49].

A distância D_E (*distância Euclideana*) entre dois pixels $p = (x_p, y_p)$ e $q = (x_q, y_q)$ é definida como:

$$D_E(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}. \tag{2.10}$$

Para calcular a transformada de distância Euclideana através da IFT, nós pensamos na imagem binária de entrada como sendo um grafo ponderado não direcionado, onde os vértices correspondem aos pixels, conforme apresentado na seção 2.1. Cuisenaire [15] mostrou que a relação de adjacência a ser utilizada deve ser escolhida com cuidado, de acordo com o tamanho da imagem de entrada, para que o cálculo do mapa de distâncias seja sempre correto. Uma escolha errada da relação de adjacência faz com que a condição ($C2$) da seção 2.3 possa ser violada. Apesar disto, a utilização de vizinhança 8 apresenta excelentes resultados em grande parte dos casos, sendo empregada em todos os exemplos apresentados neste trabalho, a não ser quando outra vizinhança está indicada.

Para que possamos utilizar a fila hierárquica de prioridades, é necessário que os valores de custo de caminho sejam todos inteiros. No entanto, como na definição de distância Euclideana aparece uma raiz quadrada, tais valores podem ser números reais quaisquer. Contornamos este problema ao utilizarmos sempre o quadrado da distância Euclideana nos cálculos. Na Figura 2.10 apresentamos um exemplo de um mapa de distância, onde os valores correspondem ao quadrado do valor da distância Euclideana de cada pixel ao

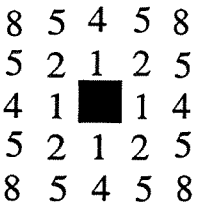


Figura 2.10: Mapa de distância, onde os valores correspondem ao quadrado do valor da distância Euclideana de cada pixel ao pixel representado em preto.

pixel preto. Já na Figura 2.11 temos um exemplo de anotação de imagem que queremos gerar após o cálculo da transformada de distância Euclideana, a partir das sementes s e s', mostradas na Figura 2.11a. Os números representados na Figura 2.11b indicam os quadrados dos valores das distâncias Euclideanas de cada pixel à semente mais próxima.

A função de custo de caminho a ser utilizada para a transformada de distância Euclideana pode ser definida como:

$$pf(P) = (\sum_{i=1}^{n-1} |x_{p_i} - x_{p_{i+1}}|)^2 + (\sum_{i=1}^{n-1} |y_{p_i} - y_{p_{i+1}}|)^2. \tag{2.11}$$

Nesta função de custo de caminhos os pesos das arestas não aparecem explicitamente. Podemos imaginar que uma aresta (p, q) tenha um custo que pode ser calculado pela diferença $pf(s, \ldots, p, q) - pf(s, \ldots, p)$. Para ver que esta função determina que os arcos possuam pesos variáveis, vamos analisar o exemplo da Figura 2.12. No exemplo, temos duas sementes, s e s', e dois pixels normais, p e q. Suponha que estejamos querendo determinar o custo da aresta (p, q) (aresta mais escura presente na figura). Se estivermos vindo em um caminho que começa na semente s, o custo da aresta (p, q) será 1, uma vez que de s a p temos uma distância Euclideana quadrática igual a 4 e de s a q igual a 5. Já se estivermos vindo a partir de s', o custo desta aresta será 5, uma vez que o quadrado da distância Euclideana de s' a p é 8 e de s' a q é 13. Vale notar que o custo do caminho calculado através da equação 2.11 não depende somente das posições do primeiro e do último pixel do caminho, como seria natural de se pensar, uma vez que estamos utilizando a soma dos valores absolutos dos deslocamentos.

A transforma imagem-floresta pode ser utilizada para resolver este problema uma vez que a função de custo de caminho 2.11 é suave. Note que apesar das arestas poderem assumir valores diferentes para diferentes prefixos de caminhos, o valor é sempre o mesmo se os prefixos forem caminhos de custo mínimo. No entanto, na forma em que esta solução está colocada, o custo requerido para computar esta função é muito alto, uma vez que temos que calcular duas somatórias para cada aresta sendo analisada. Podemos diminuir

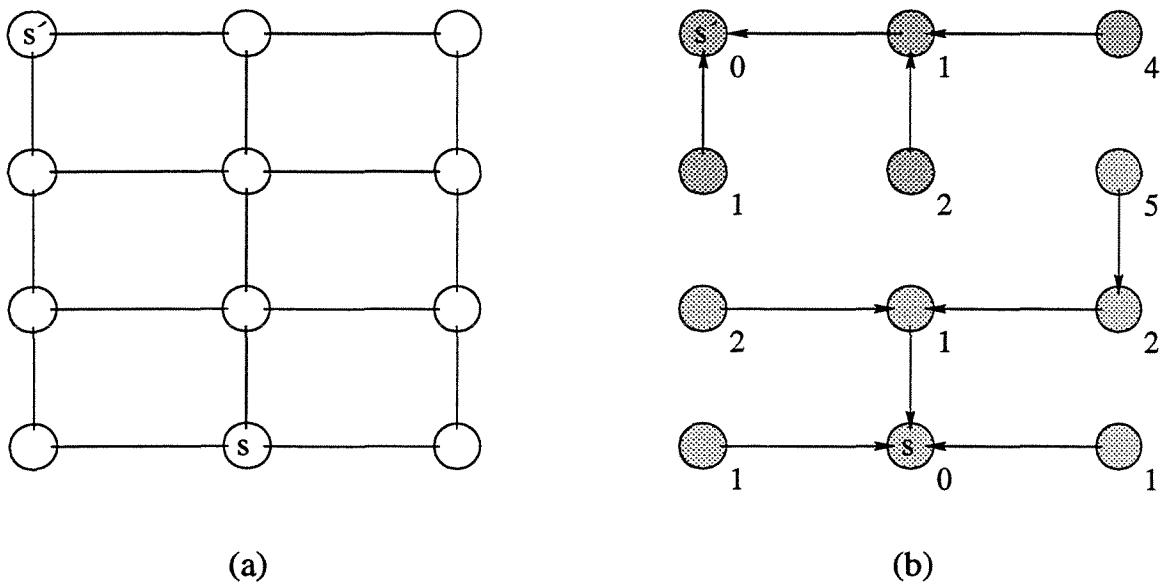


Figura 2.11: Exemplo de anotação de imagem gerada após o cálculo da transformada de distância Euclideana, a partir das sementes s e s' (a), utilizando vizinhança 4. Os números representados em (b) indicam o quadrado do valores das distâncias Euclideanas de cada pixel à semente mais próxima.

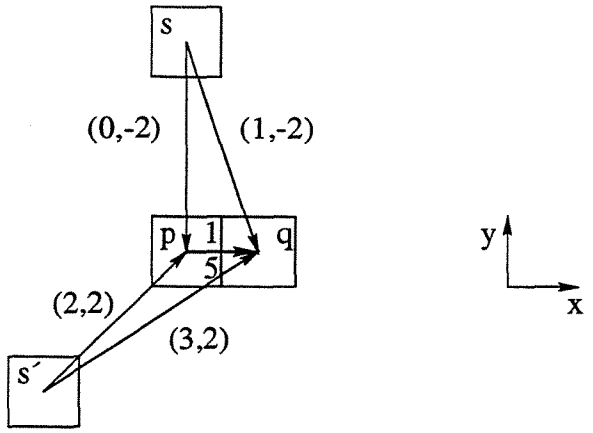


Figura 2.12: Exemplo que mostra porque as arestas do grafo devem ter pesos variáveis. Temos duas sementes no grafo, s e s' e dois outros pixels, p e q. Queremos determinar o custo da aresta (p, q). Se estivermos vindo em um caminho que começa na semente s, o custo da aresta (p, q) será 1, uma vez que de s a p temos uma distância Euclideana quadrática igual a 4 e de s a q igual a 5. Já se estivermos vindo a partir de s', o custo desta aresta será 5, uma vez que o quadrado da distância Euclideana de s' a p é 8 e de s' a q é 13.

este custo armazenando para cada pixel os vetores de deslocamentos (dx, dy) ao longo do caminho da semente mais próxima até este pixel, como mostrado na Figura 2.13. Nesta figura temos mostrados dois possíveis caminhos da semente s ao pixel p, utilizando vizinhança 8. Quando há um deslocamento horizontal, incrementamos o contador dx do pixel destino, e quando há um deslocamento vertical, incrementamos o contador dy. Desta maneira, o custo acumulado do caminho mais curto a um pixel p encontrado até um determinado instante pode ser calculado como $(dx[p])^2 + (dy[p])^2$, e não precisamos mais calcular estas somatórias a cada nova avaliação de aresta.

Assim, o algoritmo básico da transformada imagem-floresta, apresentado na seção 2.5 pode ser modificado para armazenar os vetores de deslocamento, obtendo o algoritmo descrito a seguir.

Transformada de Distância Euclideana via IFT

Entrada: Uma imagem I, uma relação de adjacência ρ, e um conjunto S de sementes rotuladas.

Saída: três imagens, *custo* (custo acumulado do caminho), *pai* (pixel pai) e *rot* (rótulos), que representam a floresta computada.

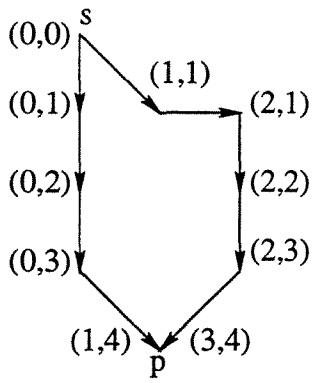


Figura 2.13: Temos mostrados dois possíveis caminhos da semente s ao pixel p, utilizando vizinhança 8. Quando há um deslocamento horizontal, incrementamos o contador dx do pixel destino, e quando há um deslocamento vertical, incrementamos o contador dy. Desta maneira, evitamos calcular as somatórias presentes na Equação 2.11 a cada nova avaliação de aresta.

Estruturas de dados auxiliares: uma fila de prioridades Q, uma lista L de pixels que já terminaram de ser processados, duas imagens dx e dy, que armazenam os vetores de deslocamento da raiz mais próxima até cada pixel e uma variável auxiliar tmp.

início

1. para todo pixel $p \in I$, atribua ∞ para $custo[p]$, $dx[p]$ e $dy[p]$, e *nil* para $rot[p]$ e $pai[p]$;
2. para todo pixel semente $s \in S$, atribua 0 para $custo[s]$, $dx[p]$ e $dy[p]$, o rótulo associado a s para $rot[s]$, e insira s em Q;
3. *enquanto* Q não estiver vazia *faça*
   a. remova o pixel p de Q tal que $custo[p] = \min_{p' \in Q}\{custo[p']\}$ e insira p em L;
   b. *para* cada pixel q vizinho de p de acordo com ρ, $q \notin L$, *faça*
      (i) calcule $tmp \leftarrow (dx[p] + |x_p - x_q|)^2 + (dy[p] + |y_p - y_q|)^2$, representando o custo do caminho para se chegar a q passando por p;
      (ii) *se* $tmp < custo[q]$ *então*
         a. faça $custo[q] \leftarrow tmp$, $rot[q] \leftarrow rot[p]$, $pai[q] \leftarrow p$, $dx[q] \leftarrow dx[p] + |x_p - x_q|$ e $dy[q] \leftarrow dy[p] + |y_p - y_q|$;

b. *se* $q \notin Q$ *então* insira q em Q *senão* atualize a posição de q em Q;

fim se;

fim para;

fim enquanto;

fim

Análise 2.6.1 Complexidade Computacional

Para se utilizar uma fila hierárquica de prioridades neste algoritmo, necessitamos determinar quantas posições (*buckets*) estarão presentes na mesma. Para tanto, precisamos saber qual é o valor máximo que o peso de uma aresta pode assumir. Vamos considerar que estamos utilizando como relação de adjacência a equação 2.1, com valor R. Neste caso, assumindo que a imagem tem largura $L+1$ e altura $H+1$, o peso máximo ocorre quando temos apenas um pixel semente, em um dos cantos da imagem, por exemplo na posição $(0,0)$, e estamos avaliando a aresta mais distante desta semente, a aresta entre o pixel $p = (L - \frac{R\sqrt{2}}{2}, H - \frac{R\sqrt{2}}{2})$ e o pixel $q = (L, H)$, conforme a Figura 2.14. Fazendo as diferenças de custo de caminho de acordo com a Equação 2.11, este peso será:

$$\begin{aligned} w(p,q) &= L^2 + H^2 - (L - \frac{R\sqrt{2}}{2})^2 - (H - \frac{R\sqrt{2}}{2})^2 \\ &= L^2 + H^2 - (L^2 - LR\sqrt{2} + \frac{R^2}{2}) - (H^2 - HR\sqrt{2} + \frac{R^2}{2}) \\ &= R\sqrt{2}(L+H) - R^2. \end{aligned} \tag{2.12}$$

Se a imagem for quadrada ($L = H$), temos que o valor dado pela Equação 2.12 é $O(\sqrt{n})$, onde n é o número de pixels presentes na imagem, levando a uma complexidade $O(n\sqrt{n})$. Desta maneira, no caso geral este algoritmo poderá não ser linear, sendo $O(n \max\{L, H\})$ com a utilização da fila hierárquica. No entanto, nos diversos experimentos que realizamos, o comportamento obtido foi linear, uma vez que esta distância máxima raramente é alcançada.

Uma característica interessante deste algoritmo é que a transformada de distância Euclideana é calculada simultaneamente no interior do objeto e no fundo da imagem, ao contrário de vários algoritmos existentes. Um exemplo de transformada de distância

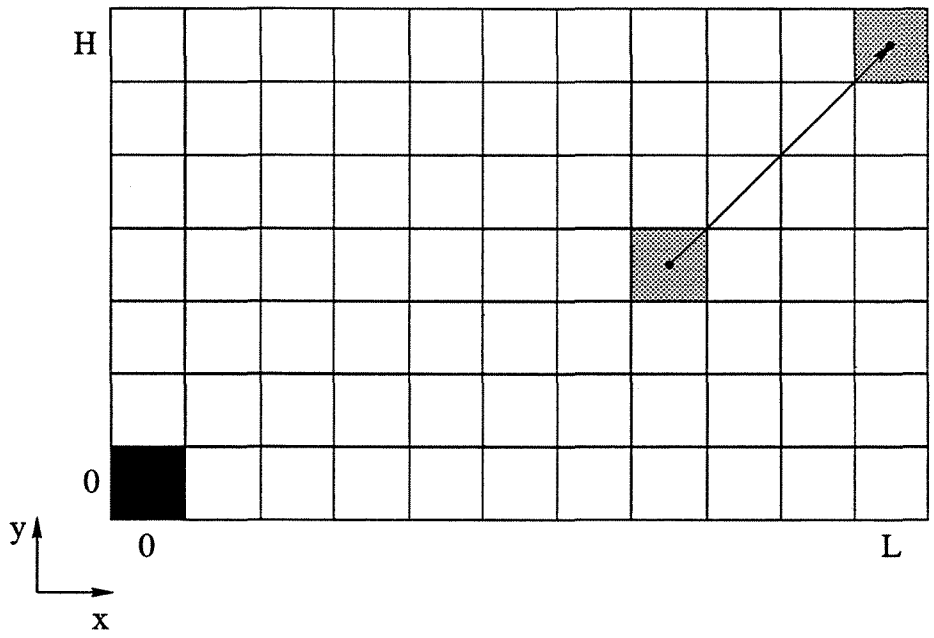
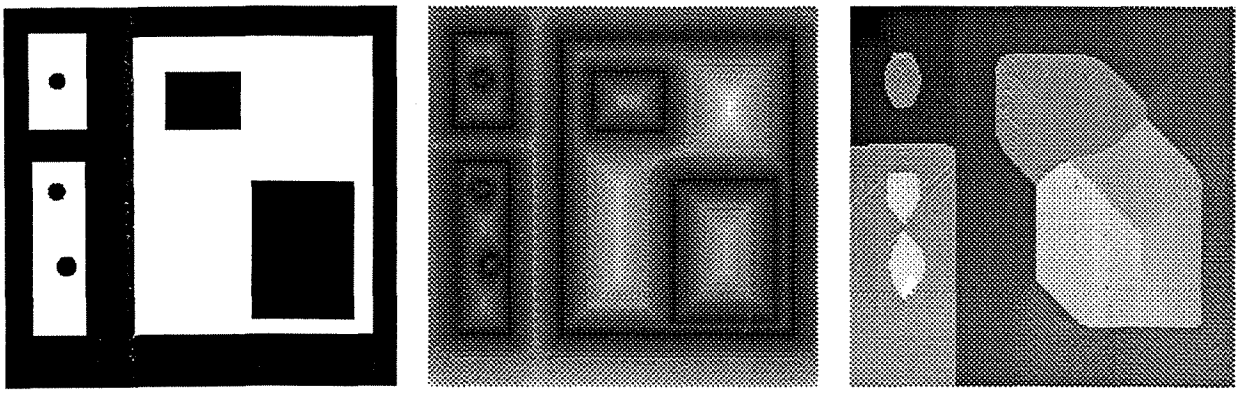


Figura 2.14: Situação onde ocorre o peso máximo de uma aresta, de acordo com a Equação 2.11, utilizando a relação de adjacência dada na equação 2.1, para um dado valor de R. Temos uma imagem de largura $L+1$ e altura $H+1$, uma semente no pixel $(0,0)$ (representada em preto), e estamos avaliando a aresta que liga o pixel $(L-\frac{R\sqrt{2}}{2}, H-\frac{R\sqrt{2}}{2})$ ao pixel (L,H).



(a) (b) (c)

Figura 2.15: Exemplo de transformada de distância Euclideana (b) para os objetos presentes em (a). Note que o algoritmo calcula o mapa de distâncias tanto dentro quanto fora dos objetos, simultaneamente. Em (c) estão representados os rótulos dos pixels, que representam as zonas de influência das sementes.

Euclideana calculada através da IFT pode ser visto na Figura 2.15. Na Figura 2.15a temos a imagem de entrada. Os pixels pertencentes às bordas dos objetos são escolhidos para serem sementes, cada um recebendo um rótulo único. Na Figura 2.15b temos a imagem de custos de caminhos, neste caso correspondendo à distância Euclideana de cada pixel ao pixel de borda mais próximo. Já na Figura 2.15c temos os rótulos associados aos pixels, que representam a partição da imagem em zonas de influência das sementes.

Tendo feito esta apresentação da transformada imagem-floresta, vamos apresentar os operadores que se enquadram mais naturalmente no modelo da IFT, que são os de segmentação de imagens.

Capítulo 3

Segmentação de imagens via IFT

No capítulo anterior mostramos que a transformada imagem-floresta é uma metodologia de resolução de problemas em processamento de imagens que consiste em reduzir o problema desejado ao cálculo de florestas de caminhos mínimos. Estas florestas representam uma partição ótima da imagem de acordo com algum critério. Desta maneira os primeiros operadores que aparecem naturalmente como candidatos a serem resolvidos através da IFT são os de segmentação de imagem.

Segmentar uma imagem significa particioná-la em regiões de interesse, permitindo a extração de objetos significativos da mesma, sendo geralmente o primeiro passo no processo de análise de imagens. Uma boa segmentação é fundamental para o sucesso de diversas aplicações, sendo um campo de estudo que tem atraído o interesse de muitos pesquisadores há vários anos.

O processo de segmentação pode ser automático ou interativo, podendo ser visto como a combinação de duas atividades: a identificação dos objetos de interesse e a delineação destes objetos. A identificação é a tarefa que indica a localização aproximada do objeto, enquanto a delineação é a tarefa que define a extensão do objeto na imagem. Enquanto o computador realiza muito bem a tarefa de delineação, o processo de identificação de objetos em imagens genéricas é muito difícil. Na verdade, esta é a razão pela qual a segmentação automática é considerada uma das atividades mais difíceis em processamento de imagens [35]. Desta forma, se tornou interessante o desenvolvimento de métodos que permitem ao usuário auxiliar o computador na tarefa de identificação dos objetos, gerando métodos interativos.

Os métodos de segmentação podem ser divididos aproximadamente em duas classes:

1. métodos baseados em bordas;

2. métodos baseados em regiões.

Os métodos da primeira classe extraem objetos através da identificação e perseguição das bordas dos mesmos. A identificação das bordas pode ser feita de maneira automática ou interativa, enquanto que a perseguição pode ser feita através de um processo de minimização do custo de se percorrer a borda [33, 54]. Outros métodos que estão nesta classe são os baseados em contornos ativos. Nestes, temos inicialmente um contorno, que passa a ser deformado pela ação de duas forças, uma interna, que está relacionada com características do próprio contorno, e uma externa, relacionada a propriedades da imagem. Um objeto é dado então pela região delimitada pelo contorno, quando a energia deste é minimizada [12, 42]. Já na segunda classe estão os métodos que realizam a classificação de acordo com regiões da imagem. Entre eles, estão os métodos baseados em pixels, que classificam os objetos através de propriedades individuais dos pixels, como por exemplo o brilho. Estes métodos podem funcionar quando diferentes objetos têm histogramas distintos. Assim, os objetos de interesse são extraídos através de limiarizações na imagem [35]. Também pertencem a esta classe os métodos baseados em crescimento de regiões, nos quais alguns pixels são escolhidos para serem sementes, que passam a competir pelos demais pixels da imagem, gerando regiões conexas [3, 39].

A *IFT* já foi utilizada para atacar o problema de segmentação através da perseguição de bordas através dos métodos *live-wire-on-the-fly*, *live-lane* e *3-D live-wire* [33, 34]. Também já foi apresentado um trabalho que combina contornos deformáveis com segmentos de custo mínimo [23]. Deste modo, vamos apresentar alguns métodos de segmentação baseados em regiões cuja implementação via *IFT* já foi descrita [32, 48], e acrescentar a este conjunto os operadores baseados em conexidade *fuzzy*.

3.1 Segmentação baseada em regiões

Na segmentação baseada em regiões, inicialmente é escolhido um conjunto de sementes, de maneira automática ou manual, a partir dos quais regiões se expandem até que todo pixel da imagem pertença a alguma região. As diferenças nos métodos baseados neste paradigma estão na forma com que as sementes são calculadas, na flexibilidade ou não dos algoritmos em tratar qualquer pixel como sendo semente e na maneira com que a expansão é feita, atingindo ou não uma partição ótima de acordo com algum critério.

Algumas das vantagens da utilização da transformada imagem-floresta são o fato que ela permite o desenvolvimento de diversos operadores a partir de um único algoritmo básico, através de uma escolha apropriada de parâmetros, e que as partições calculadas são ótimas, desde que a função de custo de caminho escolhida seja suave.

Vamos mostrar três métodos clássicos que podem ser desenvolvidos através da *IFT*:

- transformada linhas divisoras de águas;

- crescimento de regiões baseado em similaridade;
- conexidade *fuzzy*.

3.1.1 Transformada linhas divisoras de águas

Considere que a imagem de entrada é uma superfície feita de plástico, onde fazemos pequenos furos nos pixels sementes. A seguir mergulhamos esta superfície, com uma velocidade constante, em uma bacia com água. A água entra pelos furos nas sementes, inundando a imagem. No ponto de encontro de águas vindas de sementes distintas erguemos barragens. A transformada linhas divisoras de águas (LDA ou *watershed*) [3] calcula partições equivalentes a estas barragens.

Na abordagem tradicional, esta transformada só podia ser calculada a partir dos mínimos regionais da imagem, obtendo uma super-segmentação desta. Posteriormente permitiu-se utilizar como semente qualquer pixel da imagem, através de uma operação de mudança de homotopia da imagem, que fazia com que apenas as sementes fossem pontos de mínimos regionais [65]. A IFT permite o cálculo da LDA a partir de marcadores sem a necessidade de uma realização prévia de mudança de homotopia [48].

Apesar de podermos calcular a LDA em uma imagem qualquer, as partições resultantes são mais significativas quando calculadas em uma imagem de gradientes, uma vez que as barragens tenderão a ser erguidas em pontos onde o gradiente é alto. Para tanto, é comum a utilização de gradientes morfológicos para imagens em níveis de cinza ou coloridas [65].

Para modelar a LDA através da IFT, utilizamos a relação de adjacência vizinhança 4 ou vizinhança 8 e consideramos que o custo da aresta $a = (p, q)$ é igual ao brilho do pixel q na imagem, e que a função de custo de caminho é a aresta de peso máximo (equação 2.4).

3.1.2 Crescimento de regiões por similaridade e competição entre sementes

O crescimento de regiões por similaridade está baseado no seguinte pressuposto: a imagem pode ser particionada em regiões que possuem grande similaridade interna, com a similaridade entre pixels pertencentes a regiões vizinhas sendo baixa.

O custo de cada aresta é calculado por uma função de dissimilaridade. Particionamos a imagem em zonas de influência de pixels sementes. A zona de influência de uma semente é definida pelos pixels cuja distância a partir desta semente é mínima quando comparada com as distâncias a partir das demais sementes. Esta distância está relacionada à "dissimilaridade". Para imagens em níveis de cinza, temos que alguns exemplos de funções de dissimilaridade entre dois pixels p e q na imagem I são:

$$w(p,q) = |I(p) - I(q)|, \tag{3.1}$$

e

$$w(p,q) = K - Ke^{-\frac{(I(p)-I(q))^2}{\sigma^2}}, \tag{3.2}$$

onde σ representa uma variância admitida e K é uma constante que indica o valor máximo que uma dissimilaridade pode assumir.

Já em imagens multi-banda com d bandas, exemplos de função de dissimilaride entre dois pixels $p(x_p, y_p) = (I_1(p), I_2(p), \ldots, I_d(p))$ e $q(x_q, y_q) = (I_1(q), I_2(q), \ldots, I_d(q))$ são:

$$w(p,q) = \sqrt{\sum_{k=1}^{k=d} (I_k(p) - I_k(q))^2}, \tag{3.3}$$

e

$$w(p,q) = \max_{1 \leq k \leq d} |I_k(p) - I_k(q)|. \tag{3.4}$$

A Equação 3.1 utiliza apenas a informação de brilho dos pixels para determinar o custo das arestas. Uma vez que ela utiliza o valor absoluto da diferença dos brilhos, temos que esta é uma função simétrica. A Equação 3.2 permite que adeqüemos o custo entre pixels vizinhos de maneira a penalizar menos uma aresta se a diferença de brilho for próxima a uma diferença média ou se a variância de intensidades dentro do objeto de interesse for grande. Já a Equação 3.3 é adequada para medir a diferença entre duas cores quando elas estão representadas em um espaço de cor conveniente, como os espaços CIE Lab e CIE Luv, que são perceptualmente uniformes [71] (na prática, a raiz quadrada é calculada). A Equação 3.4 representa o valor absoluto máximo da diferença de intensidade entre os pixels considerando cada banda em separado. As possibilidades de escolha de funções de dissimilaridade são muito grandes. Poderíamos, por exemplo, definir funções que levassem em conta um valor de desvio na intensidade de pixels pertencentes a uma vizinhança de cada pixel, levando a funções que se comportem melhor em regiões de textura.

Uma vez que definimos como os custos dos arcos serão calculados, devemos escolher a função de custo de caminho a ser utilizada. A função soma dos pesos das arestas tende a penalizar regiões grandes, ainda que a similaridade entre os pixels da região seja muito alta. Já a função aresta de peso máximo é mais robusta, contornando este problema, tendo se comportado melhor nos testes feitos.

3.1.3 Conexidade *fuzzy*

A conexidade *fuzzy* (*fuzzy connectedness*) [59, 67] é um método de segmentação baseado na teoria de conjuntos *fuzzy*. Na base deste método estão relações *fuzzy* "locais" chamadas de

funções de afinidade, que quantificam quão próximos dois pixels estão e quão parecidos eles são em relação a propriedades baseadas nos brilhos de pixels presentes em vizinhanças dos pixels considerados. Uma relação *fuzzy* "global" chamada conexidade é definida baseada na afinidade. O valor de conexidade entre dois pixels p e q é dado pela maior "força" de conexidade entre todos os caminhos possíveis de p a q na imagem. A força de conexidade de um caminho é definida como sendo o valor do elo mais fraco neste caminho, ou seja, a menor afinidade entre pixels vizinhos que estejam neste caminho.

A função de afinidade é geralmente obtida através de uma etapa prévia de treinamento, sendo atribuído para cada objeto dois valores, a intensidade média dos pixels deste objeto, e a variância destas intensidades dentro do objeto. Um exemplo de função de afinidade entre dois pixels p e q de uma imagem I é

$$Ke^{-\frac{(\frac{I(p)+I(q)}{2}-\mu_i)^2}{\sigma_i^2}} \tag{3.5}$$

onde μ_i é o valor médio das intensidades dos pixels pertencentes a um determinado objeto i e σ_i é a variância destas intensidades.

A segmentação pode ser realizada de duas maneiras. Na primeira, chamada de conexidade absoluta, temos apenas um objeto, dado por todos os pixels que possuem conexidade com algum pixel semente maior do que um certo limiar. Na segunda, chamada de conexidade relativa, vários objetos podem estar presentes, e um pixel é classificado como sendo pertencente a um objeto se a conexidade com aquele objeto é maior do que a com os demais objetos.

Vamos lembrar que a transformada imagem-floresta minimiza funções suaves de custo de caminho. No caso da conexidade *fuzzy* estamos interessados em maximizar o valor da conexidade, com a força de conexidade sendo dada pela aresta de peso mínimo no caminho. Assim, para adequar a conexidade *fuzzy* à IFT, transformamos a função de afinidade em uma dissimilaridade, pegando o complemento da mesma. Assim, para a função de afinidade dada na equação 3.5 e no caso de um único objeto, podemos associar a função de dissimilaridade dada na equação 3.6.

$$w(p,q) = K - Ke^{-\frac{(\frac{I(p)+I(q)}{2}-\mu)^2}{\sigma^2}}, \tag{3.6}$$

Desta maneira, podemos utilizar como medida de conexidade a aresta de peso máximo no caminho, dada pela equação 2.4.

A conexidade *fuzzy* absoluta pode então ser dada simplesmente por uma limiarização na função de custo de caminho. Melhor ainda, podemos propagar o rótulo de objeto somente para os pixels cujo valor de custo de caminho seja inferior a um certo limiar, acelerando o processo.

No entanto, no caso da conexidade *fuzzy* relativa surge um problema: uma mesma aresta (p, q) pode assumir valores diferentes se estivermos chegando a p a partir de objetos diferentes, ainda que o custo de caminho em p seja o mesmo para ambos os objetos. Ou seja, o uso simultâneo de valores específicos de μ_i e σ_i para cada objeto i faz com que a função de custo de caminho não seja suave. Felizmente temos uma solução simples, porém mais custosa, para tal fato: podemos combinar as afinidades relativas entre cada par de pixels adjacentes em uma única afinidade e então calcular a partição. Esta combinação é do tipo $\max_{\forall i} \alpha_i(p, q)$, onde i é o objeto e α_i é a afinidade relativa a este objeto. Em seguida, calculamos a partição utilizando as afinidades combinadas.

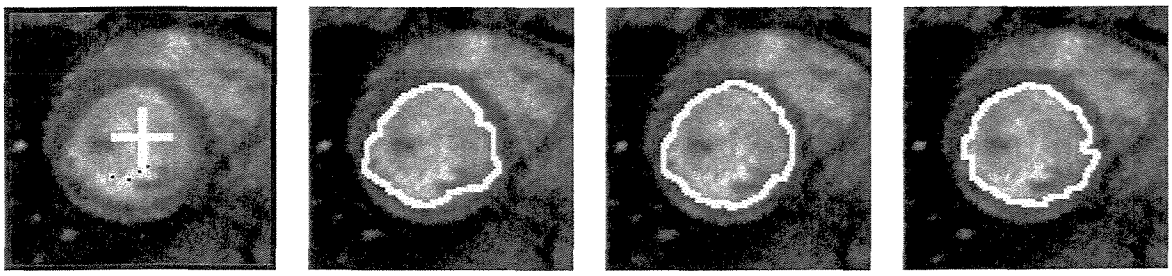
A utilização da transformada imagem-floresta para a segmentação através de conexidade *fuzzy* representa uma implementação eficiente destes operadores, uma vez que tradicionalmente eles têm sido implementados através de métodos mais lentos, baseados em programação dinâmica.

3.2 Aplicações e resultados

Ilustramos o uso dos métodos de segmentação descritos neste capítulo em imagens médicas, utilizando sementes calculadas de maneira automática. Normalmente uma filtragem adequada possibilita um resultado melhor para a segmentação. Nos exemplos apresentados foram utilizados basicamente dois tipos de filtro: filtros Gaussianos e filtros alternados seqüenciais seguidos da operação de *leveling* (ver seção 5.2.2).

Na Figura 3.1a, temos uma imagem de um ventrículo esquerdo que deve ser segmentado. Na mesma imagem estão mostradas as sementes internas (pixels sobre a cruz) e externas (pixels pertencentes à borda da imagem). Na Figura 3.1b, filtramos a imagem presente na Figura 3.1a através de um filtro alternado seqüencial, realizamos uma operação de *leveling* com estas imagens e segmentamos esta imagem filtrada a partir destas sementes através de um crescimento de regiões baseada em similaridade, utilizando a equação 3.2 como medida de dissimilaridade. Nas Figura 3.1c e 3.1d, simplificamos a imagem de entrada através de um filtro Gaussiano, e realizamos a segmentação através dos métodos conexidade *fuzzy* relativa e LDA na imagem de gradientes, respectivamente. Note que todos os resultados obtidos são aceitáveis, tendo extraído toda a região interna do ventrículo, inclusive os músculos papilares (regiões escuras no interior do ventrículo).

Na Figura 3.2a, temos uma imagem de uma córnea onde aparecem células que devem ser segmentadas. Inicialmente, aplicamos um filtro alternado seqüencial [65] para suavizar a imagem. Em seguida, calculamos os máximos regionais da imagem filtrada. Rotulamos cada componente conexa na imagem de máximos regionais com um valor distinto, e calculamos a operação dual da transformada linhas divisoras de águas (ver capítulo 5), obtendo uma partição da imagem a partir dos máximos. Na Figura 3.2b temos representados estes



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

Figura 3.1: (a) imagem do ventrículo esquerdo, com sementes dentro e fora do objeto (cruz e borda da imagem). (b) filtro alternado seqüencial + crescimento de regiões por similaridade, com pesos de arestas dados pela equação 3.2. (c) filtro Gaussiano + conexidade *fuzzy* relativa. (d) filtro Gaussiano + LDA na imagem de gradientes.

máximos, juntamente com a partição obtida. Estes elementos são utilizados como sementes para as segmentações subseqüentes, sendo os máximos regionais as sementes internas e as linhas de partição as sementes externas às células. A Figura 3.2c apresenta a segmentação obtida utilizando-se o crescimento de regiões baseado em similaridade, com os pesos das arestas sendo dado pela equação 3.3. Nas Figuras 3.2d e 3.2e temos as segmentações obtidas por conexidade *fuzzy* relativa e pela LDA calculada na imagem de gradientes, respectivamente. Note que mais uma vez todas as segmentações obtidas são razoáveis, apesar da imagem de entrada ser bastante ruidosa.

3.3 Conclusão

Apresentamos neste capítulo alguns métodos de segmentação baseados em regiões que podem ser implementados através da transformada imagem-floresta. Introduzimos pela primeira vez no ambiente da IFT os métodos baseados em conexidade *fuzzy*, que têm sido utilizados com sucesso em diversas aplicações descritas na literatura.

Apresentamos alguns exemplos de segmentação automática de imagens médicas, obtendo resultados de mesma qualidade com os diferentes métodos testados.

Tópicos interessantes para estudos futuros são a exploração de novas funções de dissimilaridade, o uso de diferentes métodos para o cálculo de gradientes e uma melhor avaliação das diferentes técnicas de simplificação de imagens que podem auxiliar na remoção de ruídos, possibilitando uma melhoria na segmentação.

No próximo capítulo apresentamos uma maneira de estender a IFT para a inclusão de mais um importante operador em análise de imagens, o cálculo de esqueletos multi-escala.

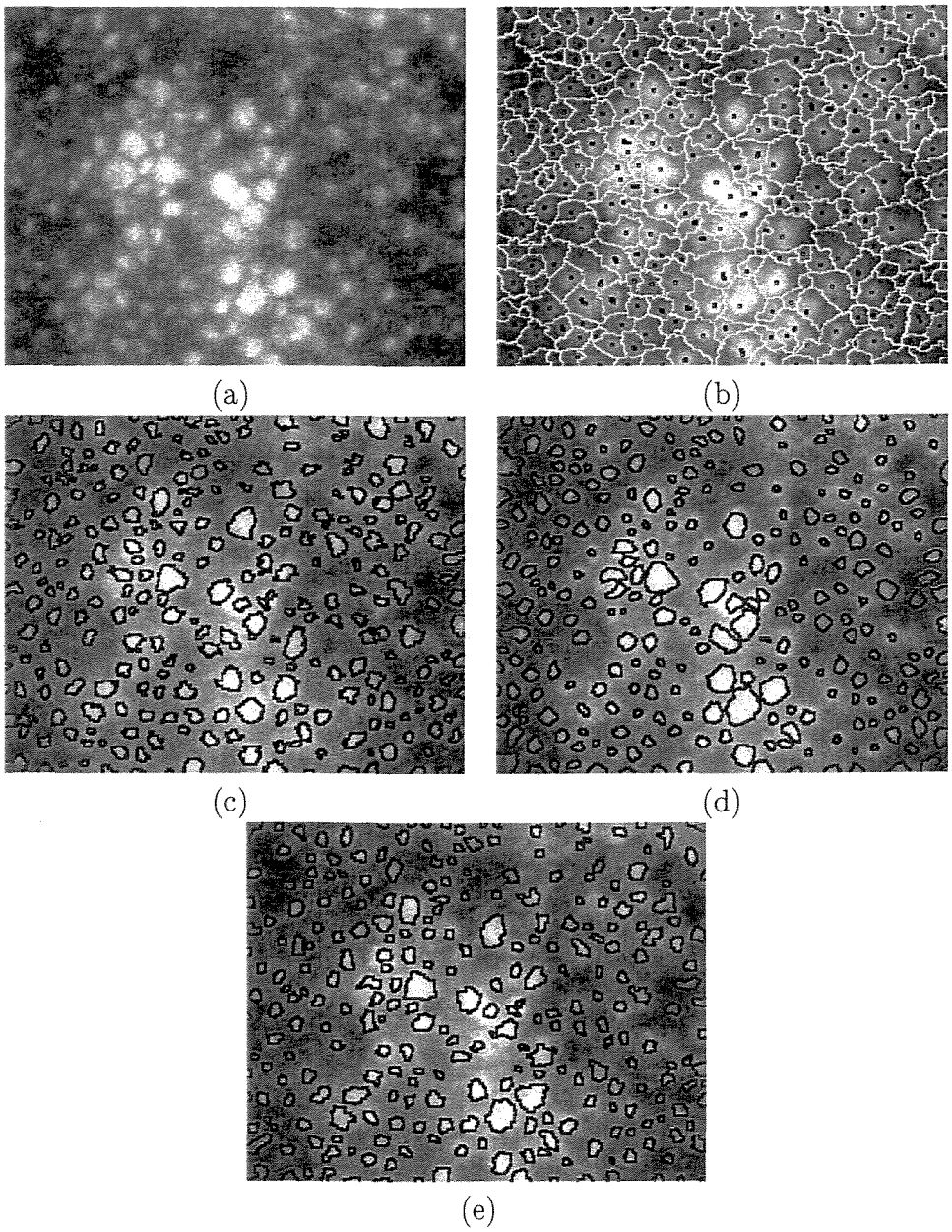


Figura 3.2: (a) imagem de uma córnea onde aparecem células que devem ser segmentadas. (b) sementes internas e externas às células. (c) similaridade utilizando equação 3.3 para pesos de arestas. (d) conexidade *fuzzy*. (e) LDA na imagem de gradientes.

Capítulo 4

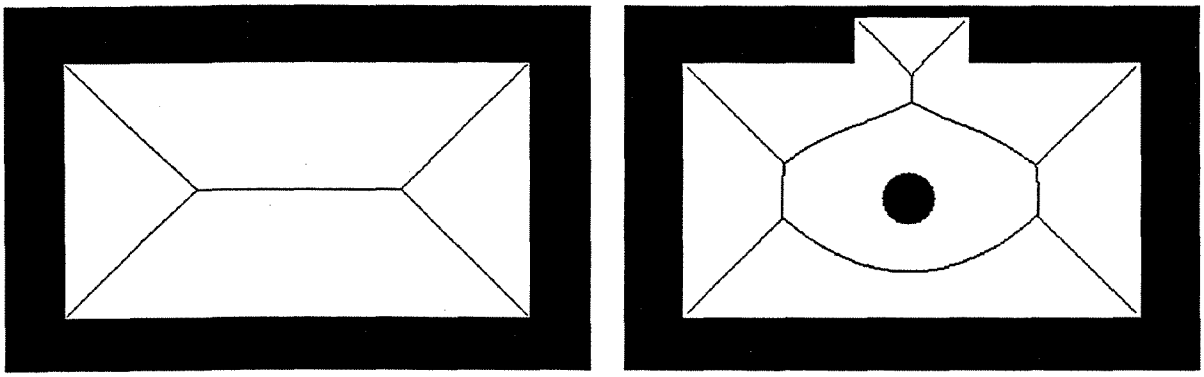
Esqueletos Multi-escala via IFT

A geração de esqueletos de objetos é uma das questões mais desafiadoras em visão computacional e análise de imagens, apesar de ser um conceito introduzido ainda em 1967, por Blum [4]. Dados objetos (ou formas), representados pelas partes brancas de uma imagem binária, podemos obter os esqueletos correspondentes. Como o próprio nome diz, esqueletos de objetos são formados pelas componentes mais essenciais da estrutura destas formas. Na falta de uma definição mais formal, podemos entendê-los como sendo a representação mais fina possível e localizada próxima às regiões centrais das diversas partes dos objetos [18] . Assim, o esqueleto de um objeto bidimensional corresponde a uma representação unidimensional do mesmo. Tal representação deve combinar informações contidas na fronteira do objeto com informações da região interna do mesmo, preservando ainda a conexidade da forma original. Desta maneira, percebemos que o esqueleto deve refletir tanto a topologia quanto a geometria do objeto.

Na Figura 4.1 temos dois exemplos de esqueletos, onde os objetos estão representados em branco e os esqueletos são as linhas pretas dentro dos objetos. Na Figura 4.1a temos um retângulo e seu esqueleto. Já na Figura 4.1b percebemos que a presença de um buraco e uma pequena mudança na forma do objeto fazem com que o esqueleto possua um ciclo e uma ramificação a mais que o esqueleto do retângulo.

Dentre as diversas operações que utilizam esqueletos, podemos citar as de agrupamento (como as de análise de documentos [40]), de segmentação [10], de planejamento de trajetórias [9] e de neuromorfometria [16, 20, 21, 41], além de descrição de forma [7, 58].

Muitos métodos para cálculo de esqueletos foram propostos, gerando esqueletos bastante distintos entre si. Os esqueletos apresentados neste trabalho seguem uma definição matemática conhecida como transformada do eixo medial (MAT), apresentada por Blum [4], que está associada à transformada do eixo de simetria (SAT). Existem, entretanto outras alternativas interessantes para a geração de esqueletos, como a baseada na teoria de campos eletrostáticos [36]. O SAT de uma forma definida em um plano contínuo



(a) (b)

Figura 4.1: Exemplo de esqueleto de objeto, onde os esqueletos estão representados nas linhas pretas dentro dos objetos. Em (a) temos que o esqueleto de um retângulo. Já em(b), a imagem possui um buraco no objeto e sua borda foi um pouco modificada, alterações que são refletidas no esqueleto.

corresponde ao lugar geométrico dos centros de círculos que satisfazem as seguintes condições:

(i) os círculos são tangentes à borda em dois ou mais pontos distintos, e

(ii) estão totalmente contidos no objeto.

O *MAT* ainda atribui a cada ponto do esqueleto o valor do raio do círculo que possui centro naquele ponto e é bitangente à borda do objeto. A Figura 4.2 ilustra alguns destes círculos, tomando como exemplo o esqueleto do retângulo da Figura 4.1a.

Apesar de haver diversos tipos diferentes de esqueletos, podemos listar como características desejáveis a todos os modelos as seguintes:

- "bom posicionamento" em relação ao objeto: o esqueleto deve mostrar simetrias no objeto, localizando-se em posições que sejam intuitivas para seres humanos. A utilização da métrica Euclideana é uma forma de se assegurar esta característica;

- invariância a transformações de corpos rígidos: quase todos os algoritmos possuem invariância a translações de objetos nas imagens; já no caso de rotações ou escalas, queremos, para objetos discretos (representados na grade), apenas que os esqueletos sejam bastante parecidos, uma vez que o próprio objeto já tem sua forma alterada ao passar por estas operações. Grande parte dos algoritmos falha até neste critério mais flexível, especialmente os que não utilizam métrica Euclideana;

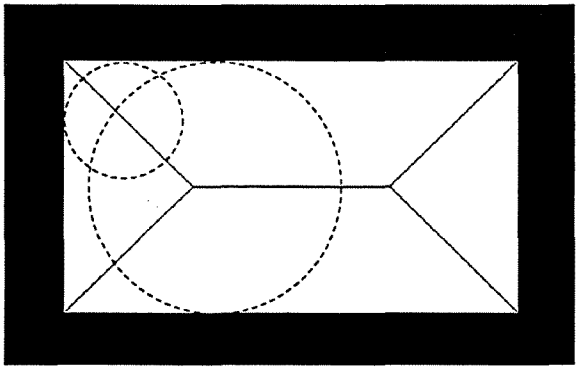


Figura 4.2: Os esqueletos do tipo SAT são formados pelo lugar geométrico dos centros de círculos que são bitangentes à forma dada e estão completamente contidos nela.

- preservação da topologia do objeto: deve mostrar a presença de buracos e manter a conexidade original do objeto;
- espessura pequena, de preferência de 1 pixel.
- baixa sensibilidade a ruídos na fronteira do objeto: todos os tipos de esqueleto são sensíveis a ruídos nas bordas dos objetos, ainda que esta perturbação afete apenas uma porção limitada do eixo medial (princípio da localidade [7]). Assim é comum que possuam rotinas de poda de esqueletos associadas. Uma alternativa bastante interessante é a possibilidade de representar os esqueletos em diversas escalas espaciais (esqueletos multi-escala), de tal forma que os ramos menos importantes vão desaparecendo ao longo das escalas;
- deve ser possível a reconstrução do objeto original a partir do esqueleto.

Muitos dos algoritmos falham em obter esqueletos com todas as características acima. Assim, apresentamos inicialmente uma breve descrição das abordagens utilizadas para a geração de esqueletos (as referências [47, 64] são bons *surveys* sobre algoritmos de cálculo de esqueletos). A seguir falamos do cálculo de esqueletos através da transformada imagem-floresta (IFT), que permite a geração de esqueletos de alta qualidade, comparável aos melhores algoritmos existentes, apresentando todas as características listadas, além de ser uma solução computacionalmente simples e eficiente. Apresentamos ainda uma prova de que os esqueletos gerados são realmente conexos em todas as escalas e um método multi-escala de filtragem de formas baseado em esqueletos. Por último, mostramos alguns exemplos de esqueletos gerados e fazemos comparações de qualidade com outros métodos.

4.1 Alguns métodos de geração de esqueletos

Muitas tentativas de se implementar a idéia original de MAT na grade discreta falharam em preservar características básicas, como a conectividade do esqueleto ou a utilização da métrica Euclideana. Deste modo, surgiram outras abordagens de cálculo de esqueletos, podendo ser classificadas, aproximadamente, nas seguintes categorias [55]:

1. simulação de "linhas de fogo" em um gramado;
2. afinamento topológico;
3. computação analítica do eixo medial;
4. extração do eixo medial a partir de um mapa de distâncias.

Vários algoritmos não se enquadram muito bem em nenhuma das classes listadas, possuindo características de duas ou mais categorias.

4.1.1 Simulação de "linhas de fogo"

Algoritmos desta classe atuam da seguinte forma: suponha que a imagem é um gramado, e que coloquemos fogo simultaneamente em todos os pontos pertencentes às bordas do objeto. Cada foco de incêndio gera uma "linha de fogo" (fronteira da propagação do incêndio), que se propaga em velocidade constante em todas as direções. O esqueleto é então formado por todos os pontos onde duas ou mais linhas de fogo chegam ao mesmo instante. Estes pontos também são chamados de pontos de extinção, uma vez que as "linhas de fogo" se extinguiriam naqueles locais.

Poucos algoritmos tentam implementar este método [47, 72], principalmente porque a grade discreta favorece a utilização de métricas regulares (como as métricas *city-block*, de *chessboard* e *chamfer*) em relação à métrica Euclideana, fazendo com que a propagação das "linhas de fogo" não seja verdadeiramente isotrópica, isto é, com que as "linhas de fogo" não sejam propagadas de maneira uniforme em todas as direções.

Em [47], os efeitos da não utilização da métrica Euclideana são reduzidos ao utilizar contornos ativos (*snakes*) para representar as "linhas de fogo". O movimento dos contornos ativos é controlado principalmente por uma transformada de distância pré-calculada. Propriedades internas da contornos ativos, tais como a resistência a dobras ou esticamentos limitam a deformação dos mesmos. Desta maneira, as "linhas de fogo" obtidas são mais suaves e são propagadas de maneira mais uniforme.

Pelo que vimos na seção 2.6.4, a transformada imagem-floresta pode ser utilizada naturalmente para simular uma propagação isotrópica de "linhas de fogo", utilizando métrica

Euclideana. Disto resultaria um algoritmo simples para a determinação de esqueletos, mas o esqueleto obtido apresentaria muitos ramos secundários. Assim, o algoritmo que propomos mais tarde é baseado nesta simulação, mas permite a construção de esqueletos multi-escala, eliminando progressivamente os ramos menos importantes do esqueleto ao se aumentar a escala.

4.1.2 Afinamento topológico

Chamamos de afinamento topológico o processo de se remover os pixels pertencentes às bordas dos objetos, sem provocar mudanças topológicas nas formas [45]. Os algoritmos desta classe analisam a importância topológica dos pixels dos objetos e não as propriedade métricas das formas. Diversas técnicas foram desenvolvidas, levando tanto a algoritmos seqüenciais [27] quanto a paralelos [38].

Estes algoritmos, apesar de gerarem esqueletos garantidamente conexos, tendem a ser muito sensíveis a ruídos, e a calcular esqueletos distorcidos, levando a resultados não intuitivos. Além do mais, a topologia das grades discretas utilizadas normalmente fazem com que este tipo de afinamento seja equivalente à utilização de métricas regulares, não Euclideanas, podendo fazer com que simples rotações produzam grandes alterações nos esqueletos.

4.1.3 Computação analítica do eixo medial

Os métodos pertencentes a esta categoria [8, 52, 56] procuram encontrar os pontos que pertencem aos esqueletos sem realizar a etapa intermediária de propagação de "linhas de fogo". Muitos dos métodos pertencentes a esta categoria computam diretamente estes pontos utilizando uma aproximação da fronteira dos objetos por polígonos. Para objetos que possuem muito ruído ou formas biológicas, como células, esta aproximação leva a uma perda muito grande de precisão. Assim, apesar de trabalharem no domínio contínuo e de acordo com a métrica Euclideana, estes algoritmos não são muito utilizados na prática.

4.1.4 Extração a partir de mapas de distância

Na quarta categoria estão os algoritmos que tentam extrair os esqueletos a partir de um mapa de distâncias pré-calculado [22, 44]. Grande parte destes algoritmos utiliza aproximações da distância Euclideana, uma vez a transformada de distância Euclideana não era calculada de maneira eficiente por algoritmos simples. Desta forma os esqueletos extraídos destes mapas tendem não ser robustos a rotações. Com o aparecimento de algoritmos que calculam a distância Euclideana exata, como os apresentados na seção 2.6.4 e em [15], o uso de mapas de distância se tornou mais interessante.

Os esqueletos obtidos através deste método são tomados como sendo as cristas do mapa de distâncias (ver Figura 2.15). Estes esqueletos tendem a ser mais precisos e suaves que os produzidos através da computação analítica ou por afinamentos topológicos [47].

A avaliação de mapas de distância está bastante relacionada com a noção de zonas de influência de pontos, sendo um problema de proximidade onde, dado um conjunto de pontos $S = \{s_1, s_2, \ldots, s_n\}$, queremos saber quais pontos (x, y) do plano estão mais próximos ao ponto s_i do que a qualquer ponto $s_k \neq s_i$ e $s_i, s_k \in S$. A solução deste problema particiona o plano em regiões delimitadas por retas, semi-retas ou segmentos de retas, chamadas de polígonos de Voronoi [57]. O conjunto destes polígonos recebe o nome de diagrama de Voronoi. Este diagrama contém toda a informação de proximidade em relação ao conjunto S, e o mapeamento deste diagrama no plano é equivalente ao computo das cristas de mapa de distâncias Euclideanas [55]. Alguns dos melhores algoritmos de geração de esqueletos são baseados no cálculo do diagrama de Voronoi, como o método de Ogniewicz [55], podendo servir como padrão de referência de qualidade de esqueleto. No entanto, estes algoritmos são computacionalmente bastante complexos, principalmente levando-se em conta que os pontos na grade digital raramente estão em posição geral [57], fazendo com que muitos casos especiais tenham que ser tratados.

4.2 Esqueletos multi-escala via IFT

Em [17] foi apresentado um método que calculava esqueletos multi-escala garantidamente conexos, com espessura 1 e baseados na métrica Euclideana para objetos simples através de dilatações exatas [18]. A implementação de tal método se mostrou, entretanto, pouco eficiente, sendo excessivamente lento até para imagens de tamanho médio.

O método apresentado nesta seção [29, 31] é uma extensão do trabalho apresentado em [17], onde, além de mantermos todas as boas características deste, permitimos o cálculo simultâneo de esqueletos de objetos com topologia arbitrária e de esqueletos por zona de influência (*SKIZ*)[46], além de possibilitar uma implementação simples e bastante eficiente, cerca de cem vezes mais rápida que a original. Podemos considerá-lo como sendo um método de simulação de "linhas de fogo", mas onde também propagamos rótulos a partir das sementes, o que nos permite a criação de diversas escalas espaciais, como será mostrado mais adiante.

Apresentamos pela primeira vez uma prova de que os esqueletos calculados pelo método são realmente conexos em todas as escalas, se considerado no plano contínuo. A prova para o caso discreto não foi atingida ainda, mas a demonstração para o caso contínuo pode nos ajudar a ter uma intuição melhor do funcionamento do método.

Uma vez que os pixels em tais esqueletos não mudam de posição nas diversas escalas espaciais, também podemos obter uma filtragem multi-escala através da reconstrução dos

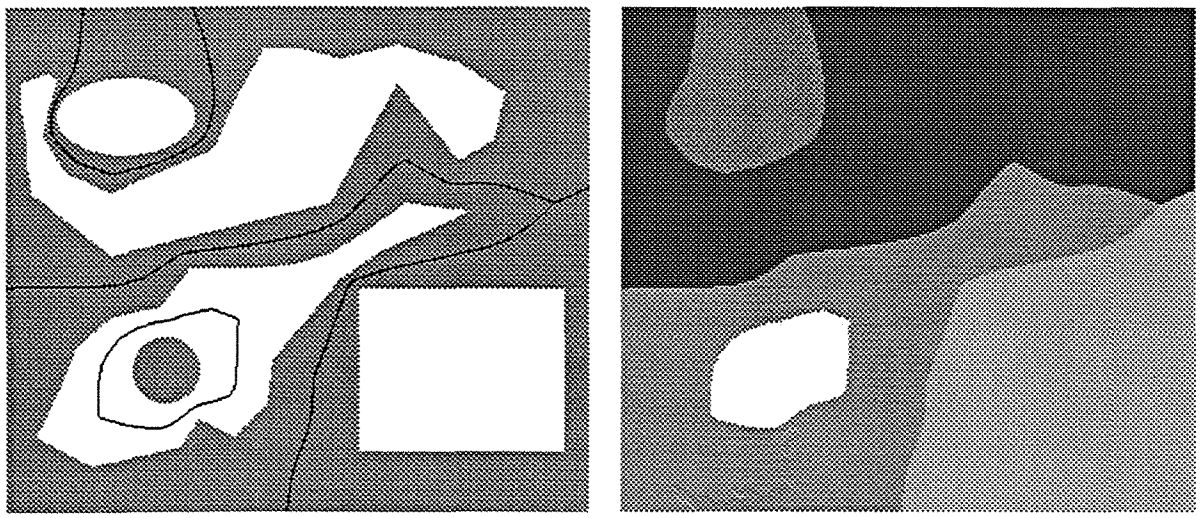
objetos a partir dos esqueletos.

A representação que propomos é formada por duas partes: o esqueleto e o $SKIZ$. O esqueleto por zona de influência pode ser definido como sendo a solução de um problema de proximidade, como o diagrama de Voronoi, mas neste caso estamos querendo saber, dado um conjunto $O = \{o_1, o_2, \ldots, o_n\}$ de objetos, quais são os pontos (x, y) da grade discreta que estão mais próximos ao objeto o_i que ao objeto o_j, com $o_i \neq o_j$ e $o_i, o_j \in O$. A distância entre um ponto e um objeto é dada pela distância deste ponto ao ponto mais próximo pertencente àquele objeto. A informação de proximidade dada pelo $SKIZ$ pode ser utilizada, por exemplo, para modelar interações entre objetos distintos.

Queremos também conseguir representar os buracos presentes nos objetos, de tal forma que esta informação esteja presente em todas as escalas espaciais do esqueleto da imagem. Para conseguir isto basta lembrarmos que na implementação da transformada de distância Euclideana através da IFT, as sementes são os pixels pertencentes às bordas (ou contornos) dos objetos. Assim, se associarmos a todos os pixels pertencentes a um mesmo contorno um rótulo que identifique este contorno, poderemos encontrar o $SKIZ$ através da detecção de transições de rótulos de contorno. Neste caso, algumas linhas do $SKIZ$ estarão dentro dos objetos, indicando a presença de buracos, enquanto outras estarão fora dos objetos, indicando o início da zona de influência de outro objeto. Na Figura 4.3a temos um exemplo de esqueleto por zona de influência ($SKIZ$, representado em preto) de quatro objetos representados em branco, um deles possuindo um buraco, enquanto que na Figura 4.3b temos os rótulos de contorno atribuídos a cada pixel pela IFT, que utilizamos para a determinação do $SKIZ$.

O outro componente de nossa representação são os esqueletos propriamente ditos. Na seção 4.1.1 dissemos que podemos simular a propagação de "linhas de fogo" através da IFT. Para calcularmos o esqueleto de objetos, associamos a cada semente um rótulo, e propagamos estes rótulos, de tal forma que os esqueletos estarão localizados nas transições de rótulos de pixel. Desta maneira, também teremos esqueletos internos e externos aos objetos, estes últimos podendo ser considerados esqueletos do fundo da imagem.

Assim, a IFT é utilizada com a formulação necessária para o cálculo da transformada de distância Euclideana, conforme descrito na seção 2.6.4, também utilizando como sementes os pixels pertencentes aos contornos dos objetos, mas propagando dois rótulos para cada pixel p ao invés de um: um rótulo de contorno ($rot_C(p)$) e um rótulo de pixel ($rot(p)$). O rótulo de contorno de um pixel p indica qual o contorno fechado mais próximo de p na imagem, e é utilizados para se determinar o $SKIZ$. Já o rótulo de pixel indica qual pixel, naquele contorno, está mais próximo de p, estando associado ao esqueleto. Uma imagem de diferenças entre rótulos é gerada e a partir de limiarizações nesta imagem obtemos os esqueletos multi-escala. As etapas do processo de determinação destes esqueletos, bem como da filtragem de formas baseada nos esqueletos estão representadas



(a) (b)

Figura 4.3: (a) Exemplo de esqueleto por zona de influência ($SKIZ$, representado em preto) de quatro formas, sendo uma com um buraco: o $SKIZ$ estará presente em todas as escalas espaciais dos esqueletos calculados. Em (b) mostramos os rótulos de contorno que foram propagados a partir das bordas dos objetos, cuja transição nos permite determinar o $SKIZ$.

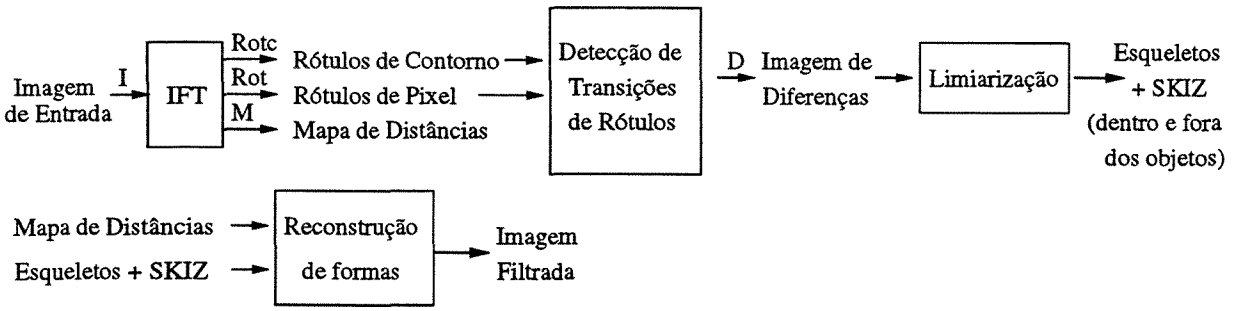


Figura 4.4: Cálculo de esqueletos multi-escala e de esqueletos por zona de influência através da IFT. A filtragem de formas é obtida a partir dos esqueletos, do $SKIZ$ e do mapa de distâncias.

na Figura 4.4.

Seja I uma imagem binária composta por um ou mais objetos, como na Figura 4.5a, onde aparecem as formas de dois neurônios localizados na retina de galinhas. Inicialmente, todo pixel do objeto que tiver pelo menos um pixel do fundo como vizinho 4 é considerado como parte de um contorno do objeto. Cada pixel deste contorno, considerado na ordem de um percurso a partir de um pixel arbitrário, recebe o mesmo rótulo de contorno $rot_C(p)$ e um rótulo de pixel $rot(p)$, sendo que se p e q são pixels consecutivos no percurso, $rot(q) = rot(p) + 1$, com rótulo de pixel do primeiro pixel do contorno sendo 1. Estes pixels são inseridos na fila Q na ordem em que são encontrados no contorno. Na prática, isto significa que se um pixel possui a mesma distância Euclideana para mais de um pixel semente, ele receberá o menor rótulo, uma vez que elementos que possuem a mesma prioridade saem da fila Q na mesma ordem em que entraram em Q. Esta regra é uma condição no método apresentado em [17], e é naturalmente satisfeita pela IFT.

A transformada imagem-floresta produz neste caso três imagens relevantes como saída: a formada pela propagação dos rótulos de contorno (Figura 4.5b), a formada pela propagação dos rótulos de pixels (Figura 4.5c), e um mapa de distâncias, que representa o quadrado da distância Euclideana de cada pixel até o pixel semente mais próximo (Figura 4.5d). O algoritmo abaixo ilustra uma implementação deste método.

Esqueletos multi-escala via IFT

Entrada: Uma imagem binária I e uma relação de adjacência ρ.

Saída: três imagens, M (mapa de distâncias Euclideanas a partir de pixels sementes, ele-

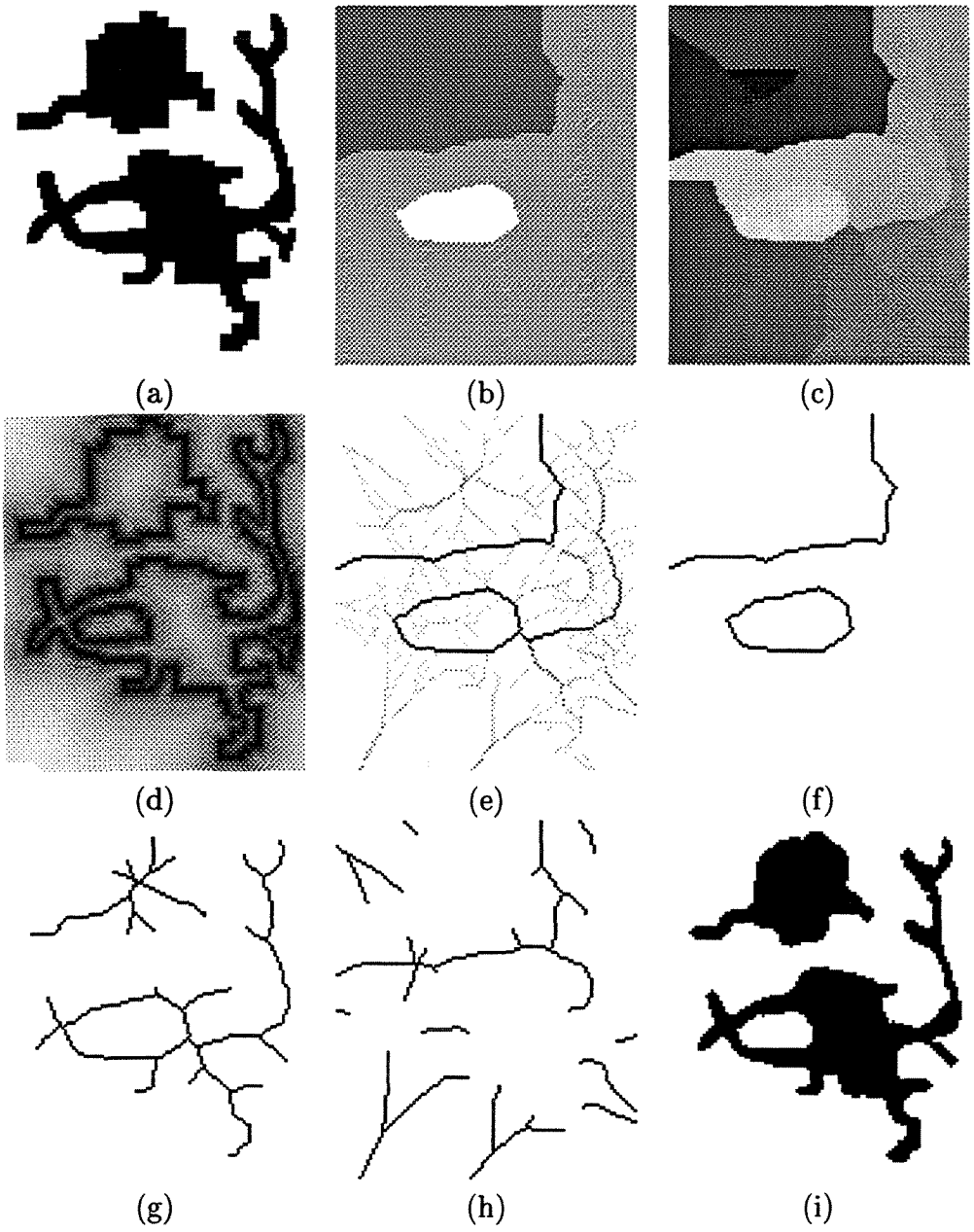


Figura 4.5: Imagens que ilustram o cálculo de esqueletos multi-escala e filtragem de formas conforme o processo mostrado na Figura 4.4. (a) Imagem binária de dois neurônios da retina de galinhas. (b)-(d) Rótulos de contornos, rótulos de pixel e mapa de distâncias Euclideanas propagadas pela IFT, respectivamente. (e) Imagem de diferenças de rótulos. (f) SKIZ. (g)-(h) Esqueletos e SKIZ dentro e fora das formas, respectivamente. (i) Imagem filtrada.

vadas ao quadrado), *rot* (rótulos de pixel) e rot_C (rótulos de contorno).

Estruturas de dados auxiliares: uma fila de prioridades Q, uma lista L de pixels que já terminaram de ser processados, duas imagens dx e dy, que armazenam os vetores de deslocamento da raiz mais próxima até cada pixel e uma variável auxiliar tmp.

início

1. para todo pixel $p \in I$, atribua ∞ para $M[p]$, $dx[p]$ e $dy[p]$, e *nil* para $rot[p]$ e $rot_C[p]$;
2. para cada contorno C presente em I, percorra cada pixel p pertencente a C, utilizando, por exemplo uma busca em profundidade em grafos [13], e, em cada um destes pixels:
   (a) atribua um inteiro consecutivo a $rot[p]$, o rótulo de contorno atual para $rot_C[p]$ e zero para $M[p]$;
   (b) insira p em Q com prioridade 0, e
   (c) atribua valor zero para $dx[p]$ e $dy[p]$.
3. *enquanto* Q não estiver vazia *faça*
   a. remova o pixel p de Q tal que $M(p) = \min_{p' \in Q}\{M[p']\}$ e insira p em L;
   b. *para* cada pixel q vizinho de p de acordo com ρ, $q \notin L$, *faça*
      (i) calcule $tmp \leftarrow (dx[p] + |x_p - x_q|)^2 + (dy[p] + |y_p - y_q|)^2$, representando o custo do caminho para se chegar a q passando por p;
      (ii) *se* $tmp < M[q]$ *então*
         a. faça $M[q] \leftarrow tmp$, $rot[q] \leftarrow rot[p]$, $rot_C[q] \leftarrow rot_C[p]$, $dx[q] \leftarrow dx[p] + |x_p - x_q|$ e $dy[q] \leftarrow dy[p] + |y_p - y_q|$;
         b. *se* $q \notin Q$ *então* insira q em Q *senão* atualize a posição de q em Q;
      fim se;
   fim para;

fim enquanto;

fim

Para gerarmos as várias escalas espaciais do esqueleto, calculamos a importância que cada pixel tem no mesmo. Como já falamos anteriormente, o esqueleto é formado nos

pontos onde ocorrem transições de rótulos de pixel. A importância de cada pixel no esqueleto é medida como sendo o comprimento (em número de pixels) do menor caminho passando somente por pixels pertencentes à borda do objeto entre os dois pixels sementes cujas zonas de influência estão se chocando naquele ponto, sendo armazenada em uma imagem de diferenças *diff* , como na (Figura 4.5e). Este mecanismo de atribuição de importância faz com que os esqueletos sejam garantidamente conexos, em todas as escalas. Para calcularmos esta imagem de diferenças, computamos primeiramente duas imagens intermediárias D_1 e D_2:

$$D_1(p) \leftarrow \max_{q \in Viz_4(p)} \{rot_C(q) - rot_C(p)\},$$

$$D_2(p) \leftarrow \max_{q \in Viz_4(p)} \{rot(q) - rot(p)\},$$

onde $Viz_4(p)$ representa o conjunto de vizinhos 4 de p. Se $D_1(p) > 0$, $diff(p) \leftarrow \infty$, significando que p pertence ao $SKIZ$ da imagem (Figura 4.5f). Caso contrário, $diff(p) \leftarrow \min(D_2(p), N - D_2(p))$, onde N representa o número de pixels do contorno mais próximo a p.

A imagem *diff* contém todas as escalas de esqueletos, mais o SKIZ. Para obter uma escala específica K, $esqueleto_K$, fazemos uma limiarização com este valor nesta imagem. Quanto maior o valor de K, menos detalhes tem o esqueleto. As definições das imagens intermediárias D_1 e D_2 garantem que os esqueletos possuem espessura 1. Se nestas expressões utilizássemos o valor absoluto das diferenças, teríamos esqueletos por zona de influência e esqueletos com dois pixels de espessura. Vamos agora provar que os esqueletos são conexos se considerarmos o caso contínuo.

Prova 4.2.1 Conexidade dos esqueletos no caso contínuo

O fato de utilizarmos o comprimento de arco (distância pela borda) entre os pixels sementes cujas zonas de influência estão se chocando como medida da importância daquele ponto de choque para o esqueleto é que garante a conexidade dos esqueletos. Primeiro, vamos notar que sempre haverá um ponto no esqueleto interno onde os valores dos choques das zonas de influência serão máximos para este esqueleto, uma vez que ele é finito. Queremos mostrar que, no caso contínuo, apenas um par de pontos semente cujas zonas de influência estão se chocando pode gerar pontos de máximo. Considere que tal par seja formado pelos pontos p e q da Figura 4.6, sendo m um ponto do esqueleto onde o valor do choque é máximo. Se considerarmos qualquer par de pontos (a, b) à esquerda do intervalo de reta que une os pontos p e q, temos que o comprimento de arco de a a b é menor

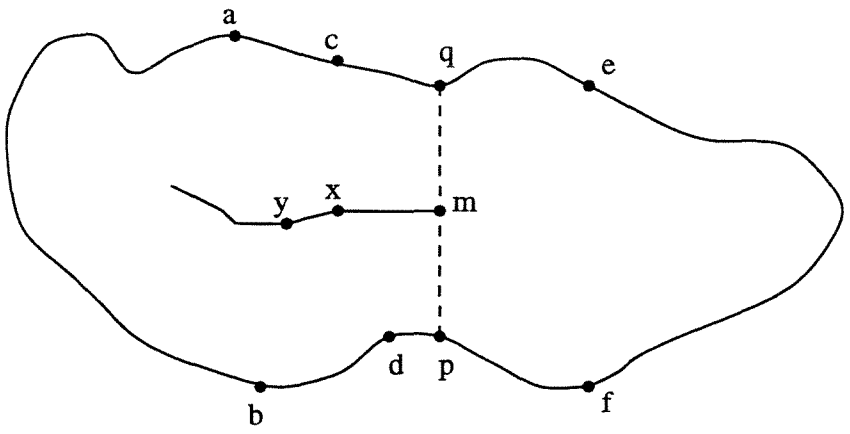


Figura 4.6: No caso contínuo, apenas um par de pontos (p, q) pode gerar pontos de esqueleto com valor máximo na imagem de diferenças. Os valores no esqueleto somente diminuem ao nos afastarmos destes pontos de máximo, o que garante que os esqueletos gerados são sempre conexos.

que o comprimento de arco de p a q, e portanto o valor presente no choque das zonas de influência de a e b é estritamente menor que o valor nos pontos de máximo gerados por p e q. O mesmo comentário vale se tomarmos um par de pontos (e, f) à direita do intervalo de reta entre p e q. Se escolhermos um ponto à esquerda e outro à direita deste intervalo, pode ser que consigamos um par de pontos (a, f) cujo comprimento de arco é maior do que o comprimento do arco entre p e q. No entanto, sabemos que as zonas de influência de p e q são vizinhas, e que zonas de influência são sempre regiões conexas. Assim, nunca poderemos ter as zonas de influência de a e f se chocando no interior do objeto. Portanto somente os pontos de esqueletos gerados pelo choque das zonas de influência de p e q são pontos de valor máximo no esqueleto.

Resta mostrar agora que ao nos afastarmos dos pontos do esqueleto onde os choques assumem valores máximos, os valores dos choques ao longo do esqueleto sempre decrescem. Considere dois pontos sobre o esqueleto do objeto, x e y. Se os pares de pontos do contorno cujas zonas de influência se chocam em y e em x forem respectivamente (a, b) e (c, d), então o valor de choque em x é maior do que o em y. Para que o valor em y seja maior do que o valor em x, devemos ter, sem perda de generalidade o par (c, b) para y e o par (a, d) par x, de tal modo que o comprimento da curva de c a b seja maior do que o comprimento da curva de a a d. Mas esta situação é impossível de acontecer, uma vez que teríamos zonas de influência – que devem ser sempre regiões conexas – se cruzando. Assim, temos que, estando x entre m e y, necessariamente o valor de choque em x é maior do que o valor de choque em y. Desta maneira, ao aumentarmos a escala em que o esqueleto está

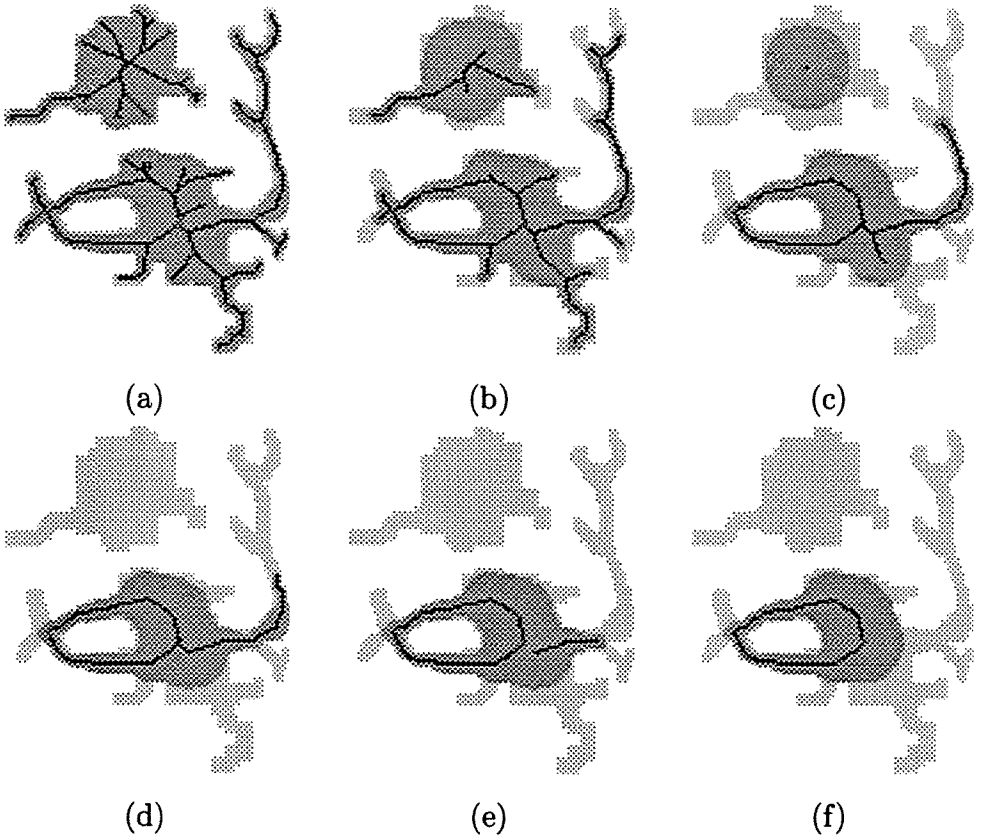


Figura 4.7: Os esqueletos conexos e de largura 1 e o $SKIZ$ dentro dos objetos estão mostrados em preto para seis escalas espaciais diferentes. A forma original está mostrada em cinza claro e a versão filtrada está mostrada em cinza mais escuro.

representado, as partes que estão mais distantes do ponto de máximo somem primeiro, e o esqueleto nunca se desconecta.

Se passarmos agora para o caso discreto, surge um problema: as regiões de influência das sementes não são mais necessariamente conexas, como nos casos em que a transformada de distância Euclideana necessita de vizinhanças maiores do que vizinhança 8. Assim, a prova de garantia de conexidade no caso contínuo não se aplica diretamente no caso discreto. No entanto, em todos os testes realizados, o esqueleto se manteve conexo em todas as escalas, o que nos leva a acreditar que eles realmente possuem esta propriedade, conforme afirmado em [17].

Uma vez que os esqueletos internos e externos aos objetos são calculados simultaneamente, podemos obter os esqueletos e $SKIZ$ internos fazendo-se uma máscara com a imagem binária de entrada (Figura 4.5g). Fazendo um mascaramento com o inverso da

imagem de entrada, obtemos os esqueletos e $SKIZ$ externos (Figura 4.5h).

A partir do mapa de distâncias e da imagem $esqueleto_K$, podemos obter uma versão filtrada dos objetos originais, pintando círculos centrados em cada pixel do esqueleto com raios dados pela distância daquele pixel até o pixel semente mais próximo (Figura 4.5i). Variando o valor de K, obtemos uma filtragem multi-escala para as formas dos objetos. Estas filtragens têm a propriedade de que as bordas em um nível k nunca passam as bordas presentes em um nível menor que k, sendo uma filtragem anti-extensiva.

Na Figura 4.7a-f mostramos em seis escalas diferentes o esqueleto e o $SKIZ$ internos de dois neurônios, cuja forma original está mostrada em cinza claro. Em cinza escuro, mostramos a filtragem da forma nas mesmas escalas. Até a Figura 4.7d, temos que alguns pontos do esqueleto se sobrepõe a pontos do $SKIZ$. Note que o esqueleto se separa do $SKIZ$ na Figura 4.7e, mas permanece conexo e com espessura 1 em todas as escalas.

Na próxima seção apresentamos exemplos de esqueletos, fazemos uma comparação de qualidade com outros métodos e apresentamos algumas aplicações dos esqueletos em neuromorfometria e imagens médicas.

4.3 Exemplos e Aplicações

Os algoritmos de cálculo de esqueletos costumam apresentar diversos problemas como falta de conexidade (esqueleto morfológico), problemas com posicionamento do esqueleto no objeto, invariância a rotações e escalas (algoritmos de afinamento), pouca flexibilidade em relação aos objetos que podem ser tratados (algoritmo baseado em teoria de campos eletrostáticos [36] só trata formas que sejam curvas simples, sem buracos) e dificuldade de implementação (como os baseados no diagrama de Voronoi).

O cálculo de esqueletos multi-escala através da IFT é simples, eficiente (tempo de execução comparável com o método de Ogniewicz[1], que é baseado no diagrama de Voronoi), gera esqueletos conexos com espessura 1, se aplica a objetos com topologia arbitrária e é baseado na métrica Euclideana, sendo bastante estável em relação a translações, rotações e escalas.

Inicialmente mostramos a qualidade dos esqueletos computados quando comparados a outros métodos, sua quase-invariância a rotações e escalas e a qualidade das filtragens multi-escalas que podem ser obtidas a partir da reconstrução das formas utilizando os esqueletos e os valores do mapa de distância. Depois apresentamos aplicações dos esqueletos em imagens médicas e em neuromorfometria, e fazemos uma comparação de eficiência com o método original[17].

[1] O programa para gerar esqueletos multi-escala utilizando o método de Ogniewicz está disponível em http://hrl.harvard.edu/people/postdocs/rlo/rlo.dir/rlo-soft.html

4.3.1 Qualidade dos esqueletos e filtragens computadas

A Figura 4.8 mostra os esqueletos de uma bicicleta calculados por quatro métodos distintos: nas Figura 4.8a e b, temos o esqueleto morfológico e o afinamento morfológico [2], respectivamente; na Figura 4.8c o esqueleto calculado pelo método de Ogniewicz [55] e na Figura 4.8d o esqueleto calculado pelo método baseado na IFT. O primeiro método gerou um esqueleto sem ser conexo, gerando uma representação pobre do objeto. O segundo é conexo, mas possui muitos ramos não podados e espessura maior do que um pixel, além de não ser muito suave em alguns trechos. Os outros dois métodos geraram boas representações do objeto, apesar de aparecerem em (c) alguns ramos indesejáveis (seta vermelha). A qualidade dos dois últimos métodos é claramente superior à dos dois primeiros, possuindo ainda a vantagem de serem métodos multi-escala, com algoritmos de poda progressiva naturalmente integrados a eles. A IFT ainda possui a vantagem de permitir uma implementação muito mais simples que o método de Ogniewicz, além de gerar esqueletos de qualidade comparável na grade discreta, em tempo de execução bastante próximo.

Outra característica importante do método apresentado é sua capacidade para filtrar formas, em diversas escalas, sem que as bordas do objeto em uma determinada escala passe as bordas do mesmo objeto em escalas inferiores, correspondendo a imagem original à escala inicial. Isto quer dizer que podemos filtrar formas de maneira anti-extensiva. Métodos lineares tradicionais para filtragem multi-escala costumam não respeitar as bordas do objeto, fazendo com que elas se tornem cada vez mais difusas. As Figuras 4.9a-c mostram o processo de filtragem multi-escala de uma forma complexa com muitas ramificações, em três escalas espaciais distintas, através do método proposto. A forma original é mostrada em cinza claro, e as formas filtradas em cinza escuro. Detalhes pequenos na forma desaparecem primeiro, enquanto as partes mais estáveis das bordas do objeto não são deslocadas, mantendo a conexidade original. Nas Figuras 4.9d-f, um filtro Gaussiano com máscara de tamanho 13 é aplicado no mesmo objeto, com diferentes valores de desvio e um mesmo valor de limiarização. As imagens filtradas não respeitam as bordas da imagem original, chegando a se tornarem desconexas em algumas escalas.

Já na Figura 4.10 apresentamos o esqueleto calculado para duas rotações (Figuras 4.10a e b) e uma escala (Figura 4.10c) do objeto mostrado em cinza claro na Figura 4.9. Em todos os casos, a limiarização que determina a escala espacial foi calculada automaticamente como sendo 5% do valor máximo presente na imagem de diferenças. Note o alto grau de similaridade entre os esqueletos mostrados.

Na Figura 4.11a temos três folhas bastante parecidas, mas em rotações diferentes. As Figuras 4.11b-d apresentam os esqueletos calculados por afinamento morfológico, pelo método de Ogniewicz e por nosso método, respectivamente. Na Figura 4.11b percebemos que o esqueleto possui dois pixels de espessura, e notamos uma tendência ao apareci-

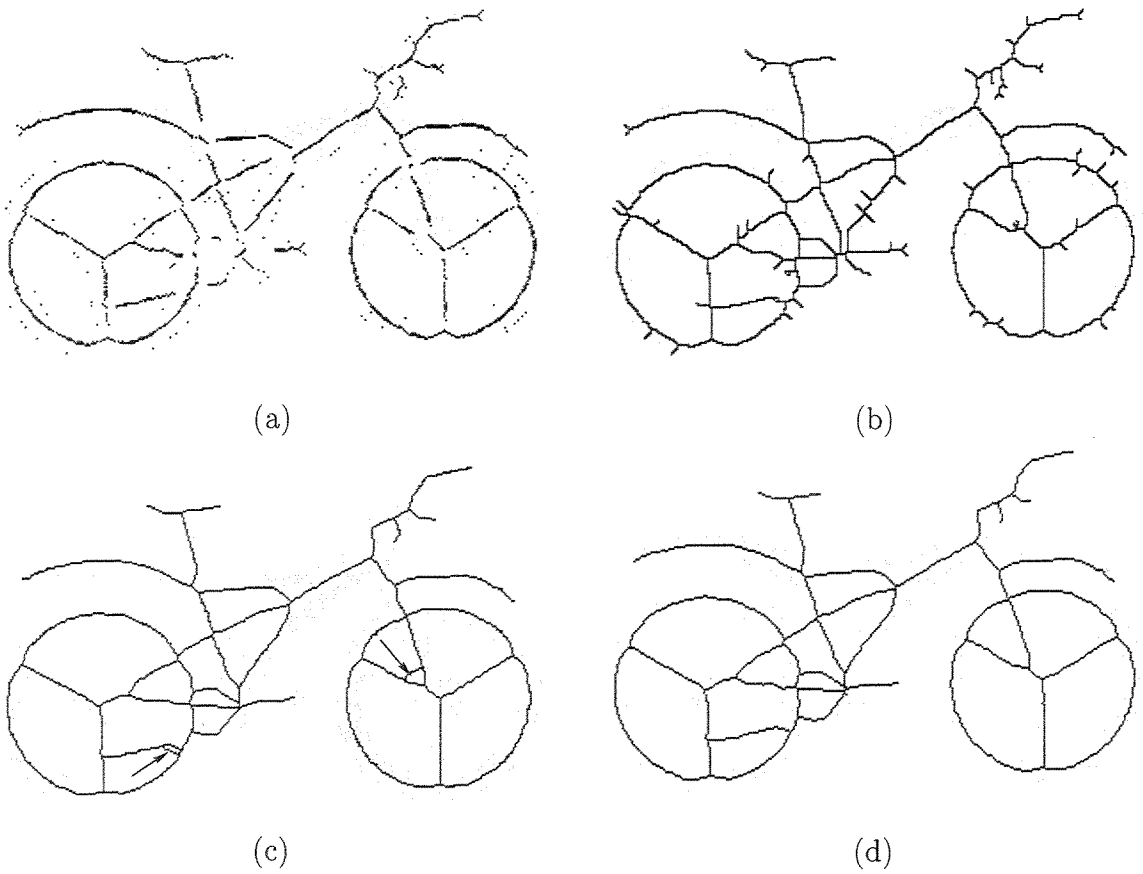


Figura 4.8: Em (a), temos o esqueleto morfológico, em (b) o afinamento morfológico, em (c) o esqueleto calculado pelo método de Ogniewicz e em (d) está representado o esqueleto calculado por nosso método. O primeiro método gerou um esqueleto sem ser conexo, gerando uma representação pobre do objeto. O segundo é conexo, mas possui muitos ramos não podados e espessura maior do que um pixel. Os outros dois métodos geraram boas representações do objeto, apesar de aparecerem em (c) alguns ramos indesejáveis, apontados pelas setas vermelhas.

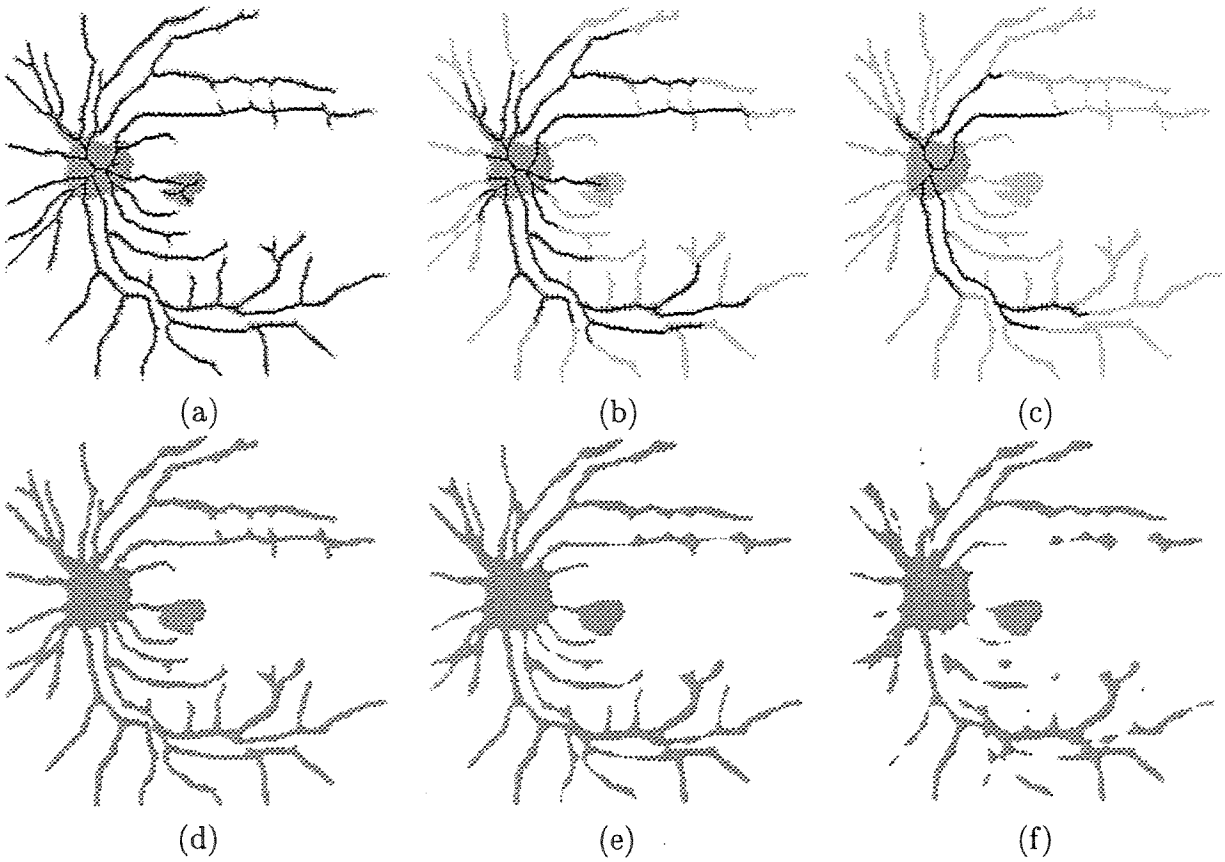


Figura 4.9: (a), (b) e (c): Filtragem multi-escala de uma forma complexa com muitas ramificações, em três escalas espaciais distintas. A forma original é mostrada em cinza claro, e as formas filtradas em cinza escuro. Detalhes pequenos na forma desaparecem primeiro, enquanto as partes mais estáveis das bordas do objeto não são deslocadas, mantendo a conexidade original. (d), (e) e (f): um filtro Gaussiano é aplicado no mesmo objeto de maneira a determinar escalas diferentes de filtragem. As imagens filtradas não respeitam as bordas da imagem original e se tornam desconexas em algumas escalas.

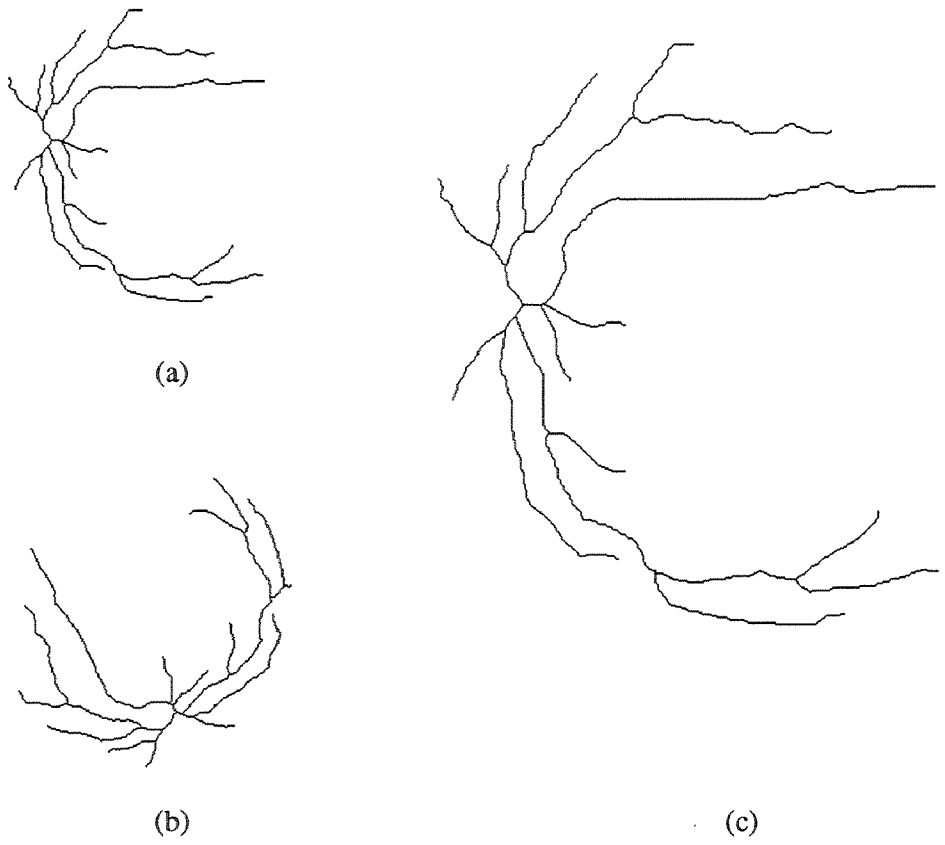


Figura 4.10: Esqueletos da forma apresentada na Figura 4.9 computados para diferentes rotações - (a) e (b) e escala (c). Em todos os casos a escala espacial determinada automaticamente

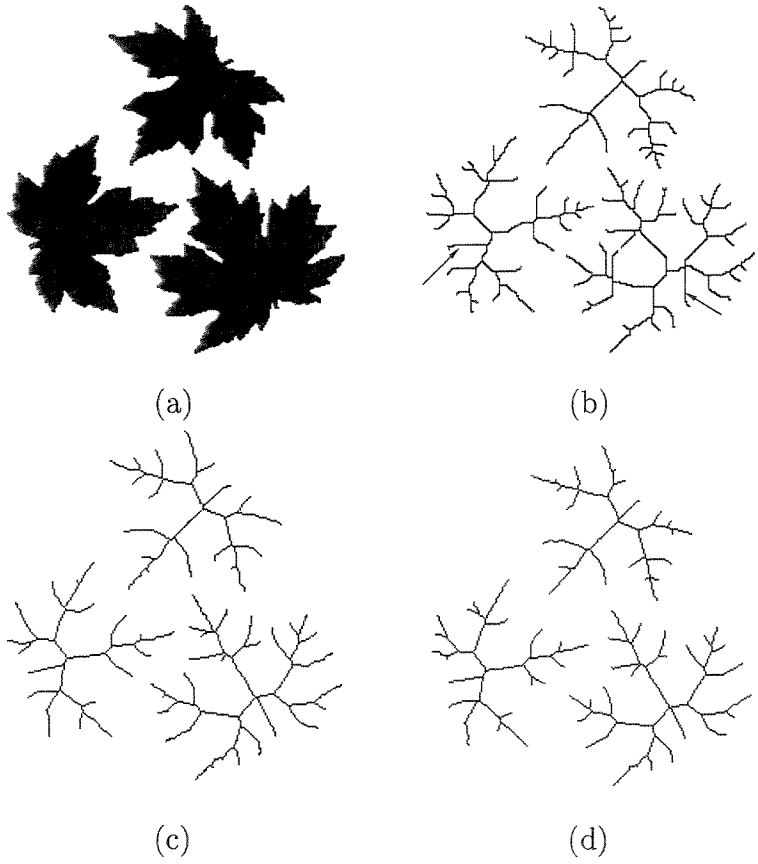


Figura 4.11: Em (a), temos uma imagem de três folhas bastante parecidas, mas em rotações diferentes. Em (b), (c) e (d), temos os esqueletos calculados por afinamento morfológico, pelo método de Ogniewicz e por nosso método, respectivamente. Em (b) percebemos que o esqueleto possui dois pixels de espessura, e notamos uma tendência ao aparecimento de muitos ramos nas posições horizontais, verticais e diagonais, indicando problemas em rotações, como nos ramos apontados pela setas vermelhas.

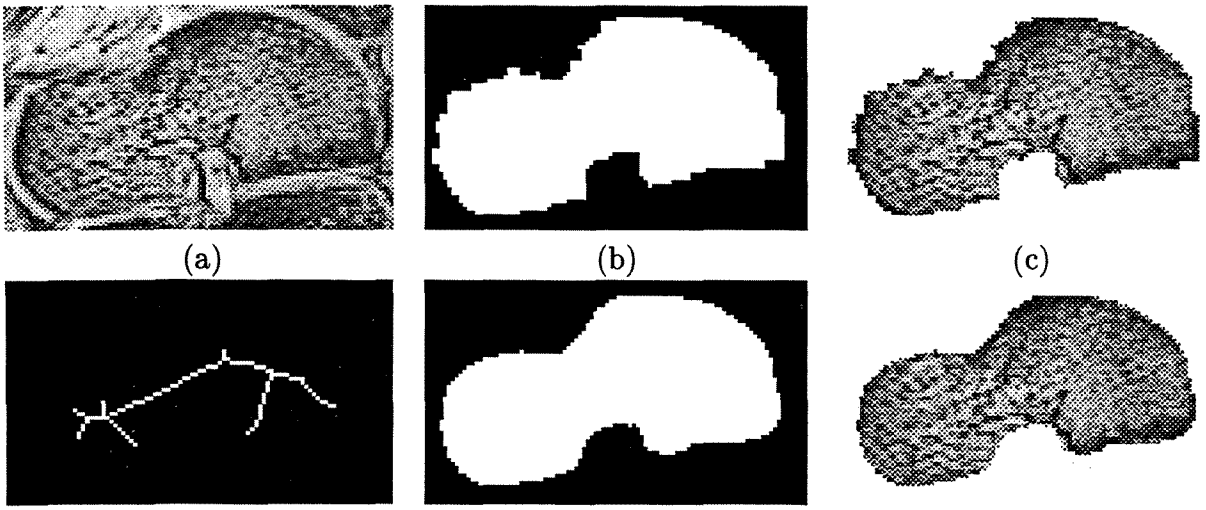


Figura 4.12: (a) Uma fatia de ressonância magnética de um pé onde a fronteira do osso talus é perseguida, produzindo uma máscara bastante recortada (b), e uma segmentação grosseira (c). (d)-(e) mostram o esqueleto e a filtragem baseada no esqueleto da máscara, produzindo um resultado mais suave. (f) mostra o resultado final da segmentação.

mento de muitos ramos nas posições horizontais, verticais e diagonais, mostrando que o esqueleto calculado equivale a um que seria calculado através de uma métrica regular. Aparecem ainda alguns ramos em posições muito pouco intuitivas, como o apontado pela seta vermelha, mostrando o tipo de problemas que este método apresenta quando objetos sofrem rotações. Os esqueletos apresentados nas Figuras 4.11b e c são bem mais suaves, não sofrendo problemas com rotações.

4.3.2 Aplicação em imagens médicas

Além de servirem em aplicações como registro de imagens e identificação de objetos, mostramos aqui a utilização da filtragem baseada em esqueletos para um refinamento de uma forma segmentada.

Muitos algoritmos de segmentação produzem bordas muito recortadas, contrariando o que esperaríamos para alguns objetos. Como exemplo, tomemos osso talus (Figura 4.12a, localizado no calcanhar, uma máscara (Figura 4.12b) e o resultado da segmentação (Figura 4.12c). Nas Figuras 4.12d e 4.12e temos o esqueleto e a reconstrução a partir do esqueleto da máscara da Figura 4.12b, e na Figura 4.12f o resultado da segmentação, com bordas mais suaves.

4.3.3 Aplicações em neuromorfometria

Os seres vivos reagem aos estímulos ambientais. As mudanças nas condições do ambiente, como sons, choques, calor e frio, são percebidos pelo organismo, que reage de acordo com o estímulo. O tecido nervoso é o responsável pela recepção e discriminação do estímulo, tendo como um de seus elementos constituintes as células nervosas ou neurônios.

Um neurônio é basicamente uma célula especializada em receber, conduzir e transmitir sinais. Possui formato alongado, apresentando três partes fundamentais: o corpo celular, os dendritos e os axônios. Enquanto o corpo celular é o centro onde estão localizados o núcleo e diversas estruturas do citoplasma, os dendritos e os axônios são estruturas alongadas, com diversas ramificações, especializadas na recepção de sinais vindos de outras células (dendritos), ou na condução de sinais do corpo celular até regiões mais distantes, transmitindo os mesmos a vários destinos, de maneira simultânea (axônios). Assim, um sinal vindo de axônios de um neurônio vizinho é captado pelos dendritos, passa pelo corpo celular e é transmitido pelos axônios, atingindo outras células. A configuração dos dendritos e axônios, como comprimento, número e estrutura das ramificações, atua como fator determinante da funcionalidade de um neurônio. Além disto, o espaço entre os axônios de uma célula e os dendritos de outra é denominado espaço sináptico, ou simplesmente sinapse. Informações como o tamanho do espaço sináptico e como os neurônios estão dispostos também são importantes para que entendamos o processo de funcionamento dos tecidos nervosos, ao determinar a interação entre células.

Avanços recentes em análise de imagens têm permitido um crescimento nas pesquisas que buscam correlacionar a forma de neurônios com sua função. Podemos agora caracterizar e quantificar aspectos geométricos de células neurais e suas estruturas. Para entendermos como os neurônios interagem com o ambiente e com suas células vizinhas, é necessário caracterizarmos corretamente a geometria das células, em especial os padrões de divisão de seus ramos, bem como as relações espaciais entre células vizinhas. Chamamos esta caracterização de neuromorfometria [19, 20, 21] . O método de cálculo de esqueletos multi-escala apresentado neste capítulo fornece uma maneira natural e bastante efetiva de se abordar este problema, bem como de agregar muitas informações a respeito de um grande número de células (*data mining*).

Considere a Figura 4.13, que foi obtida através de microscopia ótica a partir de fatias de tecido localizado na retina de galinhas, que é composto basicamente por células planares. Como estas células representam o estágio final de processamento de sinais luminosos na retina dos vertebrados, é importante estudarmos como elas estão organizadas no espaço. Além do mais, a complexidade das ramificações dos dendritos está relacionada com a sensibilidade das células.

Para abordarmos estas questões geométricas, inicialmente segmentamos estes neurônios

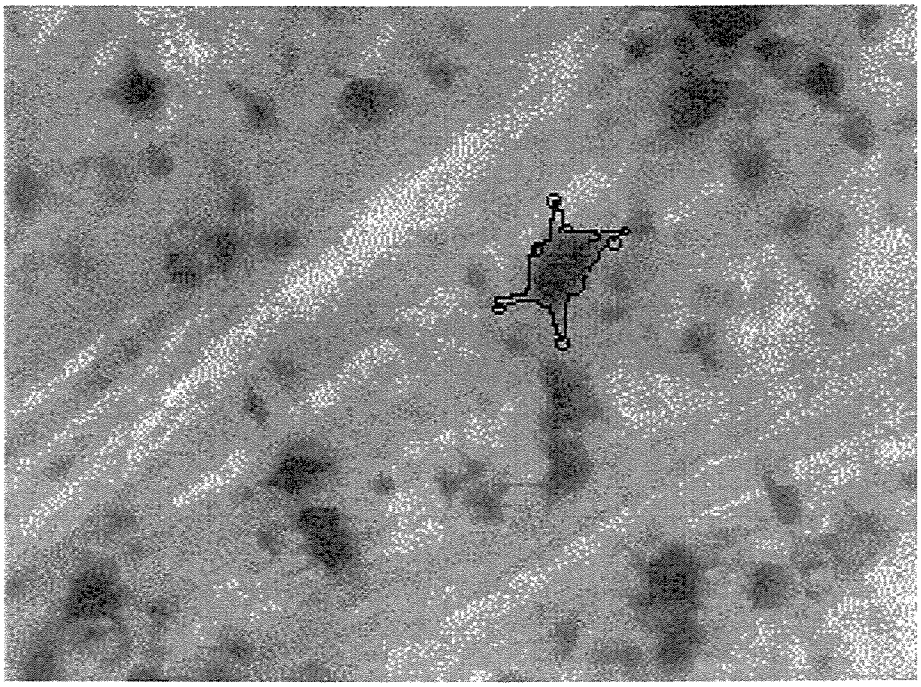


Figura 4.13: Uma imagem em níveis de cinza obtida por microscopia ótica a partir de fatias de tecido localizado na retina de galinhas. Na figura é mostrada a segmentação de um neurônio, através do método *live-wire-on-the-fly*.

da imagem, através do método *live-wire-on-the-fly*[2][33], que, embora necessite de interações com o usuário, permite uma separação efetiva das células (com suas ramificações) do fundo bastante ruidoso da imagem. A Figura 4.14 mostra, em cinza, os neurônios segmentados, bem como o esqueleto por zona de influência (em preto) que particiona a superfície da retina, e os esqueletos internos e externos dos neurônios em cada uma das regiões mostradas na imagem. Podemos perceber que o padrão de ramificação de cada célula é refletido nos esqueletos internos, permitindo a obtenção imediata de seus respectivos dendrogramas [16], uma estrutura de dados que descreve a célula, utilizada pela maioria dos métodos de simulação neural.

Além de permitir uma partição do espaço da retina e uma caracterização das ramificações dos dendritos e axônios, podemos utilizar o método de cálculo de esqueletos baseado na IFT para realizarmos uma filtragem multi-escala dos neurônios da Figura 4.14, como mostrado na Figura 4.15. Com este procedimento, podemos remover pequenos detalhes da forma dos neurônios, sem deslocar as bordas das regiões mais estáveis. Esta represen-

[2]Os neurônios foram segmentados com auxílio do programa de demonstração deste método, disponível no endereço http://www.mmorph.com/prontomask

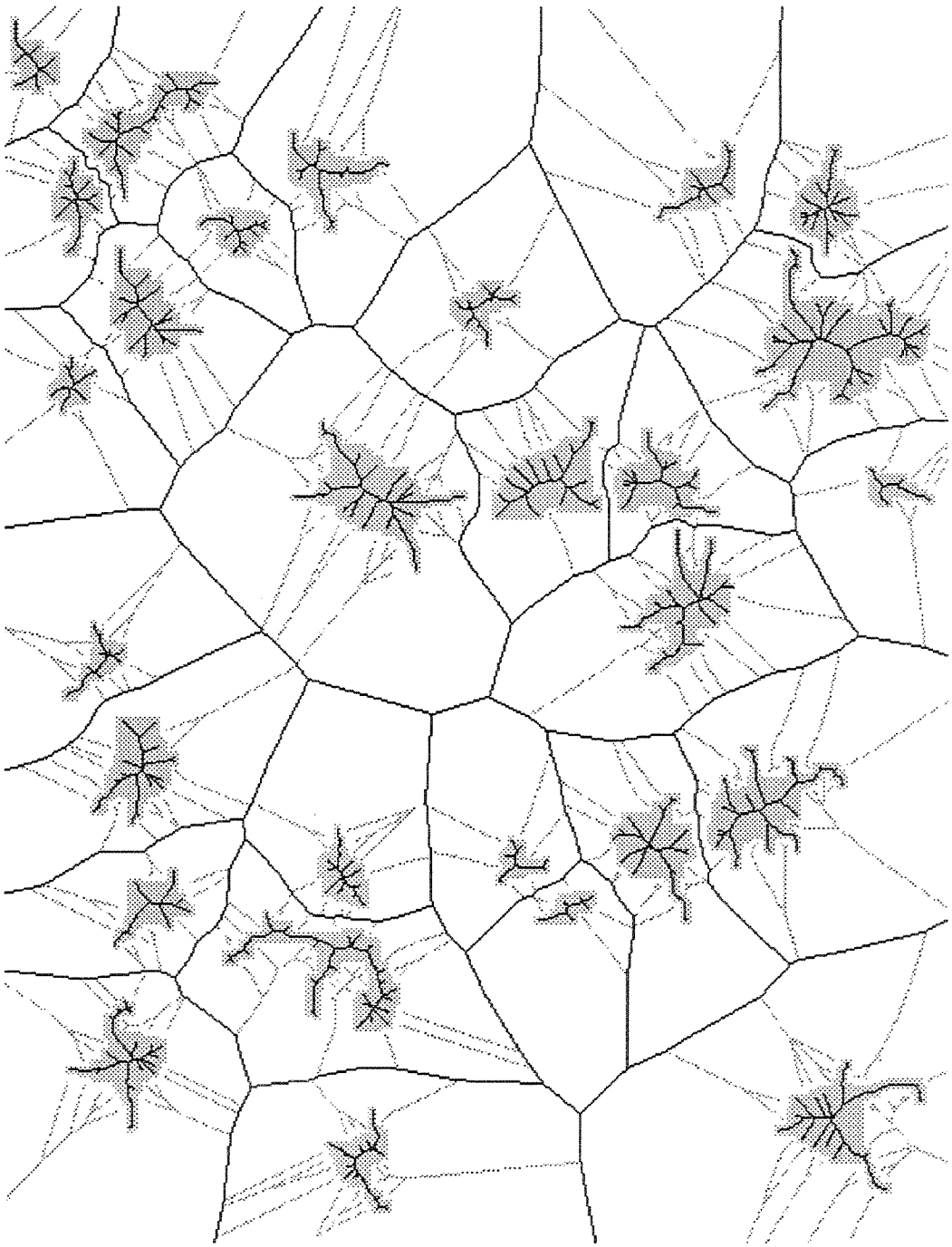
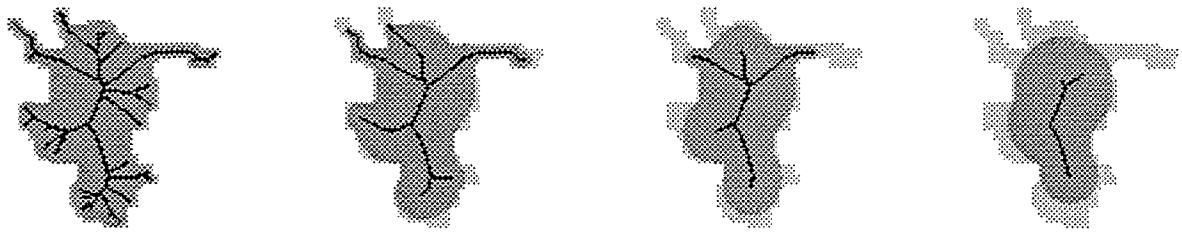


Figura 4.14: Imagem segmentada mostrando esqueletos e $SKIZ$ dentro e fora dos neurônios, extraída da Figura 4.13. O $SKIZ$ e os esqueletos internos estão mostrados em preto; os esqueletos externos e as formas originais dos neurônios em cinza.



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

Figura 4.15: Uma representação multi-escala de uma célula nervosa. A forma original é mostrada em cinza claro, as formas filtradas em cinza escuro e os esqueletos em preto.

tação permite ainda identificar o corpo de cada célula neural, que é uma estrutura neural particularmente importante.

Desta forma, o cálculo de esqueletos multi-escala via IFT se revela uma poderosa e eficiente ferramenta de auxílio à neuromorfometria.

4.3.4 Comparação de eficiência com o método original

Dissemos que o método original de cálculo de esqueletos multi-escala [17] é muito menos eficiente do que a versão baseada na transformada imagem-floresta. Para avaliarmos quão mais rápida é a nossa versão, escolhemos quatro imagens de neurônios com formas diversas, e aumentamos o tamanho das imagens, indo de 100x100 pixels a 300x300 pixels. Na Tabela 4.1 temos os tempos médios para cada imagem, em segundos, com os valores à esquerda em cada coluna correspondendo aos tempos levados pelo método original, e os à direita, aos levados pela IFT. Enquanto utilizando-se o método original os tempos crescem muito rapidamente, chegando a levar mais de dois minutos em alguns casos, o tempo máximo que a IFT precisou foi em torno de meio segundo. Os testes foram executados em um PC AMD Athlon 600Mhz.

4.4 Conclusão

Apresentamos neste capítulo a extensão da transformada imagem-floresta para o cálculo de esqueletos multi-escala, a partir de uma extensão do trabalho apresentado em [17]. Acrescentamos àquele trabalho a geração de $SKIZ$ entre objetos, a possibilidade de se trabalhar com objetos de topologia arbitrária, além de permitir um cálculo de esqueletos que em alguns casos se mostrou mais de duzentas vezes mais rápido do que o método ori-

Tabela 4.1: Tempos médios de execução (em segundos) levados para se calcular esqueletos multi-escala de quatro neurônios de formas distintas, em imagens de tamanhos diversos, utilizando o método original [17](valores à esquerda em cada coluna) e o método baseado na IFT (valores à direita), em uma mesma máquina.

tamanho da imagem	neurônio 1	neurônio 2	neurônio 3	neurônio 4
100x100	005s — 0.08s	004s — 0.04s	006s — 0.04s	003s — 0.05s
140x140	010s — 0.10s	009s — 0.07s	009s — 0.08s	005s — 0.09s
180x180	016s — 0.13s	015s — 0.16s	015s — 0.12s	008s — 0.13s
220x220	027s — 0.34s	026s — 0.35s	026s — 0.34s	013s — 0.33s
240x240	032s — 0.28s	029s — 0.28s	031s — 0.28s	016s — 0.30s
300x300	142s — 0.62s	138s — 0.62s	141s — 0.63s	073s — 0.60s

ginal. Além do mais, apresentamos uma prova de que os esqueletos gerados são realmente conexos em todas as escalas, pelo menos no caso contínuo.

Comparamos a qualidade destes esqueletos com outros métodos e percebemos que ela é equivalente à alcançada pelos melhores algoritmos de geração de esqueletos, como o método de Ogniewicz [55] (baseado no diagrama de Voronoi), sendo o algoritmo apresentado muito mais simples e com velocidade comparável.

Apresentamos ainda um operador de filtragem multi-escala, que remove pequenos detalhes da forma, sem provocar um deslocamento de bordas nas partes mais estáveis da mesma. Mostramos exemplos que comprovam a alta qualidade dos esqueletos computados e aplicações em imagens médicas e em neuromorfometria.

Um trabalho interessante e desafiador é a extensão deste método para o cálculo de esqueletos multi-escala de objetos tridimensionais, que encontraria muitas aplicações na área de imagens médicas.

No próximo capítulo estendemos a IFT para o desenvolvimento de operadores conexos, que permitem, por exemplo, o desenvolvimento de filtros que não introduzem bordas falsas na imagem.

Capítulo 5

Operadores Conexos via IFT

5.1 Introdução

Definimos uma componente conexa de uma imagem binária como sendo uma região conexa maximal pertencente ao fundo da imagem (pixels com valor 0) ou ao objeto (pixels com valor 1). Podemos estender este conceito para imagens em níveis de cinza, chamando de zona plana ou platô de uma imagem cada região conexa maximal da imagem onde todos os pixels possuem o mesmo brilho.

Segundo Salembier [61], o primeiro operador conexo que surgiu na literatura foi a abertura por reconstrução binária. Estes operadores atuam de maneira independente em cada componente conexa da imagem binária, eliminando algumas componentes sem alterar as demais.

Esta abordagem foi generalizada para imagens em níveis de cinza, passando a ser chamada de reconstrução morfológica, ou simplesmente de reconstrução [69, 70]. Os operadores de reconstrução têm por entrada duas imagens: uma máscara (I) e uma imagem marcadora (J), definidas no mesmo domínio, e tais que $I \leq J$ (ou $J \leq I$), ou seja, para todo pixel p pertencente ao domínio da imagem, $I(p) \leq J(p)$ (ou $J(p) \leq I(p)$). Tais operadores calculam a "reconstrução" da máscara a partir da imagem marcadora, obtendo uma simplificação da máscara. Esta imagem simplificada, além de ser um objeto de interesse, pode auxiliar na detecção de propriedades da imagem (como mínimos e máximos regionais), podendo ser utilizada em muitas tarefas de segmentação e de filtragem não linear de imagens.

Filtros por reconstrução foram identificados como pertencentes a uma classe maior de operadores, chamados de operadores conexos, que interagem com a imagem através de seus platôs [61, 62]. Um operador $\mathcal{O}$ é dito conexo se e somente se, para quaisquer pixels p e q pertencentes a uma mesma zona plana da imagem I, p e q também estão em uma mesma zona plana em $\mathcal{O}(I)$. Como estes operadores fazem com que algumas zonas planas

se fundam, obtemos uma simplificação da imagem onde bordas falsas não são criadas, um problema que é muito comum em diversos filtros lineares. Esta importante propriedade motivou o surgimento de diversos operadores conexos como abertura por área [69], filtros baseados em dinâmica [37], em volume [68], em complexidade e movimento [60].

Na Figura 5.1b temos um exemplo de atuação de um operador conexo sobre a imagem representada na Figura 5.1a. Note que os platôs indicados pelas letras B, D e E foram agregados a outros platôs, e que não foram adicionadas bordas falsas aos objetos.

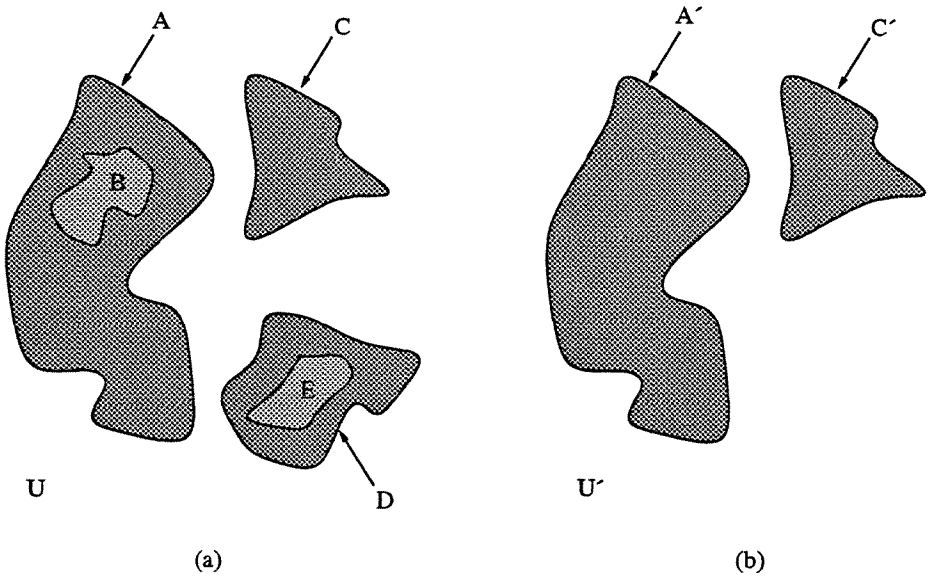


Figura 5.1: Exemplo de operador conexo. Note que a filtragem é realizada através da união de platôs da imagem de entrada.

Os operadores conexos baseados na Transformada Imagem-Floresta que serão apresentados possuem a interessante característica de calcular a transformada linhas divisoras de água a partir de marcadores [3] ao mesmo tempo em que a imagem é simplificada. Apresentaremos inicialmente os operadores básicos baseados na IFT, depois mostraremos como obter alguns outros operadores a partir destes operadores básicos, e por fim alguns exemplos de aplicação em imagens médicas e em vídeo digital.

5.2 Operadores Conexos via IFT

Nesta seção utilizamos o conceito de decomposição por limiarização de uma imagem em níveis de cinza para apresentar dois operadores: a poda de domos e o preenchimento de bacias. Podemos considerar uma imagem em níveis de cinza como sendo um relevo, onde

o brilho do pixel corresponde à altura do relevo naquele pixel. Desta forma, partes claras podem ser consideradas domos, enquanto as partes mais escuras podem ser consideras bacias. O operador poda de domos simplifica uma imagem em níveis de cinza ao reduzir a altura de seus domos, enquanto que o operador preenchimento de bacias reduz a profundidade de suas bacias.

Para permitir o desenvolvimento de operadores conexos utilizando a transformada imagem-floresta, acrescentamos a cada semente uma nova informação: o nível em que ela se encontra, chamado de nível de referência. Este nível pode ser visto como o custo inicial de qualquer caminho que parte daquela semente. Diferentes sementes podem ter diferentes custos. Como veremos adiante, a especificação destes custos é que determina o grau de simplificação obtido.

5.2.1 Poda de domos e preenchimento de bacias

Dada uma imagem $I(p)$ que associa um valor inteiro no intervalo $[0, K]$ a cada pixel p, uma pilha de imagens binárias $T_k(p)$, $k = 0, 1, \ldots, K$, pode ser criada ao atribuirmos para cada imagem binária T_k o valor 1 (objeto) para todos os pixels p nos quais $I(p) \geq k$, $k \in [0, K]$, e valor 0 (fundo) para os outros pixels. Este processo de criação da pilha de imagens binárias é chamado de decomposição por limiarização . Um exemplo de decomposição por limiarização de uma imagem em níveis de cinza (Figura 5.2a) está representado na Figura 5.2b.

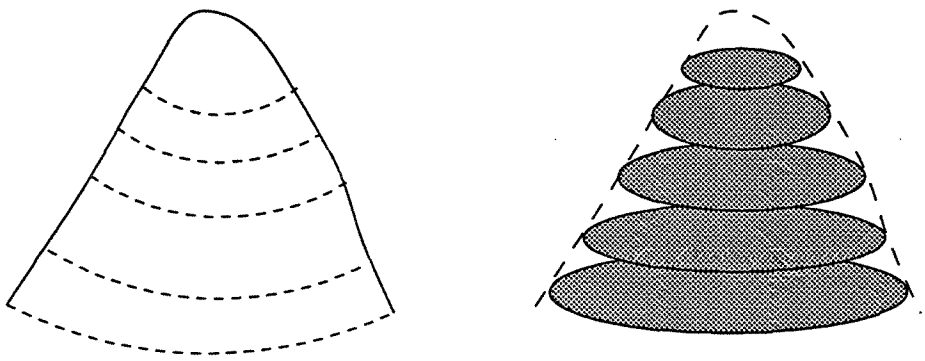


Figura 5.2: Decomposição por limiar de uma imagem em níveis de cinza

O operador poda de domos remove componentes conexas com valor 1 da pilha de imagens binárias, enquanto o operador preenchimento de bacias adiciona componentes

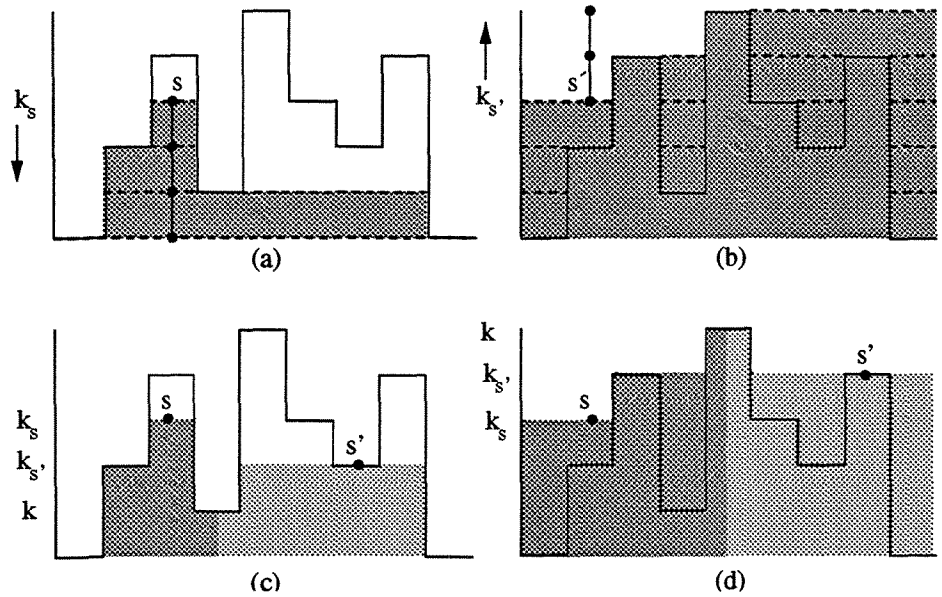


Figura 5.3: (a) Operação de poda de domos a partir de uma semente selecionada s no nível k_s. (b) Operação de preenchimento de bacias para uma semente selecionada no nível k_s. (c) Poda de domos e (d) preenchimento de bacias a partir de duas sementes s e s', nos níveis k_s e $k_{s'}$, respectivamente. Em todas as figuras as áreas sombreadas mostram as zonas de influência das sementes, e o perfil da união destas áreas representa a imagem simplificada.

conexas com valor 1 à pilha. Para simplificar a discussão, ilustramos as operações utilizando uma representação de imagens em níveis de cinza a partir de perfis unidimensionais.

A Figura 5.3a mostra uma imagem com uma semente s selecionada no nível de cinza k_s. O operador poda de domos consiste em selecionar todas as componentes conexas com valor 1 da pilha de imagens binárias que contêm s em níveis $k \leq k_s$. Este processo de "crescimento de regiões" é feito do topo à base da pilha de imagens binárias. Tais componentes são representadas pelas linhas tracejadas na Figura 5.3a e a imagem simplificada é representada pelo perfil da área sombreada na mesma figura. Note que esta operação remove todas as componentes com valor 1 que não contêm s e aquelas que contém s em níveis $k > k_s$.

A Figura 5.3b ilustra o operador preenchimento de bacias aplicado à mesma imagem, a partir se um pixel semente s', selecionado no nível de cinza $k_{s'}$. Este operador remove da pilha de imagens binárias todas as componentes conexas com valor 0 (fazendo com que passem a ter valor 1) que não contêm s' e todas aquelas que contêm s' em níveis $k \leq k_{s'}$. Este processo de "crescimento de regiões" é feito da base ao topo da pilha de imagens

binárias. Tais componentes estão representadas pelas linhas tracejadas na Figura 5.3b, e a imagem simplificada é representada pelo perfil da área sombreada na mesma figura. Note que isto equivale a fazer com que apenas as componentes conexas com valor 0 da pilha de imagens binárias que contêm s' em níveis $k > k_{s'}$ permaneçam no fundo.

Vamos supor agora que especifiquemos dois pixels sementes, s e s', nos níveis k_s e $k_{s'}$, respectivamente, como mostrado na Figura 5.3c, e que computemos a operação de poda de domos. As sementes definem zonas de influência, representadas pelas áreas sombreadas na Figura 5.3c, que se encontram no nível k onde uma componente conexa com valor 1 da pilha de imagens binárias contém as duas sementes. A imagem simplificada é representada pelo perfil da união das áreas sombreadas na mesma figura. Esta partição da imagem busca maximizar a zona de influência de cada semente. Observe que s e s' foram selecionadas em dois domos da imagem que se encontram no nível k, $k < k_s, k_{s'}$. Se s' tivesse sido selecionada em um nível $k_{s'} < k$, então a zona de influência de s invadiria e eliminaria a zona de influência de s', uma vez que o processo é realizado a partir do topo da pilha de imagens binárias. Em outras palavras, s' não seria efetiva no processo de poda de domos, sendo dominada por s. Um exemplo similar é mostrado na Figura 5.3d, com a operação de preenchimento de bacias. A imagem simplificada é representada pelo perfil da união das áreas sombreadas na mesma figura, onde as áreas sombreadas indicam as zonas de influência de cada semente. As sementes foram colocadas em duas bacias da imagem que se encontram em um nível k, $k > k_s, k_{s'}$. Se s' tivesse sido selecionada em um nível $k_{s'} > k$, então a zona de influência de s invadiria e eliminaria a zona de influência de s', uma vez que esta operação é realizada do fundo ao topo da pilha de imagens binárias.

A diferença fundamental entre estes operadores conexos e a reconstrução morfológica [70] é que, ao mesmo tempo em que geram a imagem simplificada, eles geram uma partição desta imagem simplificada. A partição resultante da aplicação do operador preenchimento de bacias corresponde à transformada de linhas divisoras de água a partir de marcadores [3] (sementes s e s'), aplicada à imagem simplificada. A mesma observação é válida para a operação poda de domos, que gera uma partição equivalente à da operação dual da transformada de linhas divisoras de água a partir de marcadores. Esta relação entre operadores conexos e a transformada de linhas divisoras de água é uma importante contribuição deste trabalho.

A implementação dos operadores de processamento de imagens usando a IFT requer a definição dos seguintes parâmetros: uma relação de adjacência, uma função de custo de caminho e um conjunto de sementes rotuladas. No caso dos operadores poda de domos e preenchimento de bacias a relação de adjacência e a função de custo de caminho são fixas para cada operador, sendo que a seleção de sementes e a atribuição de rótulos e de valores de referência definem o tipo de operação conexa a ser realizada. Na verdade estes operadores são os primeiros no contexto da IFT a requerer a especificação de valores de

referência para as sementes. Uma vez que há maneiras interativas e automáticas para a seleção das sementes e especificação de seus rótulos e níveis de referência, discutiremos estes parâmetros somente na próxima seção, nos concentrando agora nos demais parâmetros.

Dados uma imagem I, um conjunto $\mathbf{S} = \{s_1, s_2, \ldots, s_m\}$ de m pixels sementes em I com rótulos $l_{s_i} \neq 0$ e níveis de referência $k_{s_i} \in [0, K]$, $i = 1, 2, \ldots, m$, respectivamente, tratamos I como se fosse um grafo direcionado onde os pixels de I são os nós do grafo, e cada par ordenado (p, q) de pixels vizinhos 4 (ou vizinhos de acordo com alguma vizinhança maior) em I define um arco orientado que vai de p a q no grafo.

Uma vez que todos os caminhos começam em um pixel semente, definimos a função de custo de caminho da maneira a seguir. Seja P um caminho de um pixel semente s a um pixel q tal que $P = P' \cdot \langle (p, q) \rangle$ é o resultado de se acrescentar o arco (p, q) ao caminho P' no grafo. A função de custo de caminho $pf(P)$ para o preenchimento de bacias é definida como sendo:

$$pf(P) = \begin{cases} \max\{pf(P'), w(p,q)\} & \text{se } P' \neq \emptyset, \\ \max\{k_s, w(p,q)\} & \text{caso contrário,} \end{cases} \tag{5.1}$$

onde $w(p, q) = I(q)$ é o peso w de uma aresta (p, q) no grafo. Já a função de custo de caminho $pf(P)$ para a poda de domos é definida como sendo:

$$pf(P) = \begin{cases} \max\{pf(P'), w(p,q)\} & \text{se } P' \neq \emptyset, \\ \max\{(K - k_s), w(p,q)\} & \text{caso contrário.} \end{cases} \tag{5.2}$$

onde $w(p, q) = K - I(q)$ é o peso w de uma aresta (p, q) no grafo.

Em ambas as operações, a IFT computa uma floresta de caminhos mínimos onde as árvores são enraizadas nos pixels sementes definidos em $\mathbf{S}$ e cria uma anotação da imagem composta por duas propriedades para cada pixel: o rótulo propagado e o valor mínimo para a função de custo de caminho. A imagem com os valores assumidos pela minimização da função dos custos de caminho corresponde à imagem simplificada, na operação de preenchimento de bacias, e ao complemento da imagem simplificada, no caso da poda de domos. Já a imagem de rótulos corresponde à transformada de linhas divisoras de águas a partir de marcadores, no caso de preenchimento de bacias, e ao dual desta transformada, no caso de poda de domos. Ambas as operações podem ser computadas pelo algoritmo abaixo.

Poda de domos e preenchimento de bacias via IFT

Entrada: Uma imagem em níveis de cinza I, um conjunto $\mathbf{S} = \{s_1, s_2, \ldots, s_m\}$ de m pixels sementes em I com rótulos $l_{s_i} \neq 0$ e níveis de referência $k_{s_i} \in [0, K]$, $i = 1, 2, \ldots, m$,

respectivamente;

Saída: Duas imagens, $rot[p]$ e $custo[p]$, representando os rótulos propagados e os valores minimizados da função de custo de caminho, respectivamente;

Estruturas de dados auxiliares: Uma fila de prioridades Q. Uma lista L de pixels que já terminaram de ser processados. Uma variável *tmp* para o cálculo das funções de custo de caminho.

início

1. para todos os pixels $p \in I$, atribua ∞ para $custo[p]$ e 0 para $rot[p]$;
2. *para* cada semente $s_i \in \mathbf{S}$ *faça*
   a. atribua $K - k_{s_i}$ a $custo[s_i]$ se a operação for poda de domos ou k_{s_i} a $custo[s_i]$ se a operação for preenchimento de bacias;
   b. atribua l_{s_i} a $rot[s_i]$;
   c. insera s_i em Q;
3. *enquanto* $Q \neq \emptyset$ *faça*
   a. remova o pixel p de Q tal que $custo[p] = \min_{p' \in Q}\{custo[p']\}$ e insira p em L;
   b. *para* cada pixel q vizinho 4 de p, $q \notin L$ *faça*
      (i) atribua $\max\{custo[p], w(p,q)\}$ a *tmp*, onde $w(p,q) = K - I(q)$ para poda de domos e $w(p,q) = I(q)$ para preenchimento de bacias;
      (ii) *se* $tmp < custo[q]$ *então*
         a. atribua *tmp* a $custo[q]$ e $rot[p]$ a $rot[q]$;
         b. insira q em Q;
      fim se;
   fim para;

fim enquanto;

fim

O algoritmo acima executa em tempo linear se a fila Q é implementada como uma fila hierárquica de prioridades (ver Apêndice A). No algoritmo básico da *IFT*, a posição

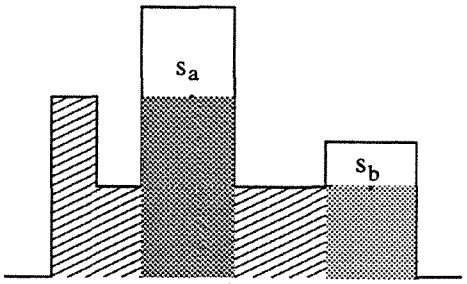


Figura 5.4: A poda local de domos é utilizada para remover da pilha de imagens binárias todas as componentes conexas com valor 1 que estão acima do nível de referência de cada semente.

de um pixel $q \in Q$ deve ser atualizada no passo 3b(ii)b, mas no caso da transformada de linhas divisoras de água tal operação não é necessária, conforme demonstrado em [48].

Uma variação muito interessante do operador poda de domos é sua aplicação regional para remover da pilha de imagens binárias somente as componentes conexas com valor 1 que estão acima do nível de referência de suas sementes, como mostrado na Figura 5.4. Nesta figura, temos duas sementes s_a e s_b, que têm por zonas de influência as regiões sombreadas. Os pixels pertencentes às regiões hachuradas nem chegam a ser processados. A imagem simplificada corresponde à união dos perfis das zonas hachuradas com as zonas sombreadas. O operador preenchimento de bacias também pode ser aplicado localmente, removendo da pilha de imagens binárias somente as componentes conexas com valor 0 (fazendo com que elas passem a ter valor 1) que estão abaixo do nível de referência de suas bacias. Chamamos estes operadores de operador poda regional de domos e preenchimento regional de bacias . Estes operadores permitem que realizemos uma filtragem regional, e é particularmente útil para rotular (extrair) componentes conexas, como veremos em exemplos mais adiante. A implementação destes operadores regionais é muito simples, uma vez que a única mudança necessária no algoritmo acima é que no passo 1, $custo[p]$ deve ser inicializado com o valor de $I(p)$ para preenchimento de bacias e $K - I(p)$ para poda de domos, para todos os pixels $p \in I$.

5.2.2 Seleção de sementes, rótulos e níveis de referência: outros operadores conexos

A seleção das sementes e de seus rótulos e níveis de referência definem o tipo de operação conexa a ser realizada. A escolha interativa de sementes constitui um novo paradigma de filtragem não-linear de imagens, merecendo ser melhor explorada em outro estudo.

Nesta seção, nos concentramos na escolha automática de sementes para mostrar como os operadores poda de domos e preenchimento de bacias podem ser combinados para implementar os seguintes operadores conexos: máximos e mínimos regionais, *h-domes* e *h-basins*, *leveling*, abertura por área, fechamento por área, abertura por reconstrução, fechamento por reconstrução, fechamento de buracos e remoção de picos [69, 70, 51].

Para facilitar a discussão, vamos definir $\mathcal{D}(I, \mathbf{S})$ e $\mathcal{B}(I, \mathbf{S})$ como sendo a aplicação dos operadores poda de domos e preenchimento de bacias à imagem I, respectivamente, a partir de um conjunto $\mathbf{S}$ de pixels sementes s com rótulos l_s e níveis de referência k_s. O algoritmo que implementa estes operadores gera como saída uma anotação da imagem, composta pelos rótulos propagados $rot[p]$ e os valores $custo[p]$ obtidos para a função de custo de caminho. Assim, dizemos que o operador $\mathcal{D}(I, \mathbf{S})$ gera uma imagem simplificada D_s, onde $D_s(p) = K - custo[p]$, e uma imagem de rótulos D_l, onde $D_l(p) = rot[p]$, para cada $p \in I$. Já o operador $\mathcal{B}(I, \mathbf{S})$ produz uma imagem simplificada B_s, onde $B_s(p) = custo[p]$, e uma imagem de rótulos B_l, com $B_l(p) = rot[p]$, para cada $p \in I$.

- Máximos regionais $\mathcal{R}_M(I)$: Calcula os platôs de I cujos níveis de cinza são maiores do que os níveis de cinza de seus vizinhos, gerando uma imagem binária R_M, onde $R_M(p) = 1$ se p pertencer a um máximo regional em I e $R_M(p) = 0$ caso contrário.

  1. Gere uma imagem X tal que $X(p) = I(p) - 1$ para todo $p \in I$;
  2. Gere uma imagem Y tal que $Y(p) = 1$ se $X(p) \geq X(q)$ para todo q vizinho-8 de p em X, e $Y(p) = 0$ caso contrário;
  3. Coloque em $\mathbf{S}$ todos os pixels $s \in Y$ tais que $Y(s) = 1$, atribua 0 para l_s (pois os rótulos não possuem significado nesta operação) e $X(s)$ para k_s;
  4. Calcule $\mathcal{D}(I, \mathbf{S})$ utilizando estes parâmetros;
  5. Gere a imagem R_M tal que $R_M(p) = I(p) - D_s(p)$ para cada $p \in I, D_s$;

- Mínimos regionais $\mathcal{R}_m(I)$: Calcula os platôs de I cujos níveis de cinza são menores do que os níveis de cinza de seus vizinhos, gerando uma imagem binária R_m, onde $R_m(p) = 1$ se p pertencer a um mínimo regional em I e $R_m(p) = 0$ caso contrário.

  1. Gere uma imagem X tal que $X(p) = I(p) + 1$ para todo $p \in I$;
  2. Gere uma imagem Y tal que $Y(p) = 1$ se $X(p) \leq X(q)$ para todo q vizinho-8 de p em X, e $Y(p) = 0$ caso contrário;
  3. Coloque em $\mathbf{S}$ todos os pixels $s \in Y$ tais que $Y(s) = 1$, atribua 0 para l_s (pois os rótulos não possuem significado nesta operação) e $X(s)$ para k_s;
  4. Calcule $\mathcal{B}(I, \mathbf{S})$ utilizando estes parâmetros;

5. Gere a imagem R_m tal que $R_m(p) = B_s(p) - I(p)$ para cada $p \in I, B_s$;

- *h-domes* $\mathcal{H}_d(I)$: Calcula uma imagem H_d que contém os domos de I cuja altura é maior que $h \in [0, K]$.

  1. Gere uma imagem X tal que $X(p) = I(p) - h$ para todo $p \in I$;
  2. Calcule $\mathcal{R}_M(X)$ e rotule cada componente conexa com brilho 1 de R_M com um valor distinto;
  3. Coloque em $\mathbf{S}$ todos os pixels $s \in R_M$ tais que $R_M(s) = 1$, atribua o rótulo da componente conexa a que s pertence para l_s e $X(s)$ para k_s;
  4. Calcule $\mathcal{D}(I, \mathbf{S})$ utilizando estes parâmetros;
  5. Gere a imagem H_d tal que $H_d(p) = I(p) - D_s(p)$ para cada $p \in I, D_s$;

  Note que os rótulos propagados (presentes em D_l) representam as zonas de influência dos máximos regionais de X, gerando uma família de partições ao se variar h. A única diferença entre entre este operador e uma abertura por reconstrução é que no segundo caso a imagem X é criada aplicando-se uma abertura morfológica a I.

- *h-basins* $\mathcal{H}_b(I)$: Calcula uma imagem H_b que contém as bacias de I cuja profundidade é maior que $h \in [0, K]$.

  1. Gere uma imagem X tal que $X(p) = I(p) + h$ para todo $p \in I$;
  2. Calcule $\mathcal{R}_m(X)$ e rotule cada componente conexa com brilho 1 de R_m com um valor distinto;
  3. Coloque em $\mathbf{S}$ todos os pixels $s \in R_m$ tais que $R_m(s) = 1$, atribua o rótulo da componente conexa a que s pertence para l_s e $X(s)$ para k_s;
  4. Calcule $\mathcal{B}(I, \mathbf{S})$ utilizando estes parâmetros;
  5. Gere a imagem H_b tal que $H_b(p) = B_s(p) - I(p)$ para cada $p \in B_s, I$;

  Note que os rótulos propagados (presentes em B_l) representam as zonas de influência dos mínimos regionais de X, gerando uma família de partições ao se variar h. A única diferença entre entre este operador e um fechamento por reconstrução é que no segundo caso a imagem X é criada aplicando-se um fechamento morfológico a I.

- *Leveling* $\mathcal{L}(I, J)$: Calcula a imagem simplificada L entre duas imagens de entrada I e J definidas no mesmo domínio. A imagem L é tal que $I(p) \leq L(p) \leq J(p)$ em regiões onde $I(p) \leq J(p)$, e $J(p) \leq L(p) \leq I(p)$ nas regiões em que $J(p) \leq I(p)$.

1. Gere uma imagem X tal que $X(p) = \min\{I(p), J \oplus e(p)\}$ para cada $p \in I, J$, onde $J \oplus e(p)$ representa a dilatação da imagem J pelo elemento estruturante e;
2. Calcule $\mathcal{R}_M(X)$;
3. Coloque em $\mathbf{S}$ todos os pixels $s \in R_M$ tais que $R_M(s) = 1$, atribua 0 para l_s e $X(s)$ para k_s;
4. Calcule $\mathcal{D}(I, \mathbf{S})$ utilizando estes parâmetros;
5. Gere uma imagem Y tal que $Y(p) = \max\{J \ominus e(p), D_s(p))\}$ para cada $p \in I, D_s$, onde $J \ominus e(p)$ representa a erosão da imagem J utilizando um elemento estruturante e;
6. Calcule $\mathcal{R}_m(Y)$;
7. Coloque em $\mathbf{S}$ todos os pixels $s \in R_m$ tais que $R_m(s) = 1$, atribua 0 para l_s e $Y(s)$ para k_s;
8. Calcule $\mathcal{B}(I, \mathbf{S})$ utilizando estes parâmetros;
9. Gere a imagem L, sendo $L(p) = B_s(p)$ para cada $p \in B_s$.

- Abertura por área $\mathcal{A}_O(I)$: Remove da pilha de imagens binárias de I (dada pela decomposição por limiarização da imagem I) todas as componentes conexas com valor 1 cujas áreas (número de pixels pertencentes à componente) for menor do que um limiar, e gera uma imagem simplificada A_O como saída. Tal operação remove os domos de I com área menor do que este limiar.

  1. Calcule $\mathcal{R}_M(I)$ e rotule cada máximo regional em R_M com um valor distinto;
  2. Para cada máximo regional em R_M, aplique localmente o operador poda de domos para calcular o nível de referência mais alto em que a componente conexa que contém o máximo regional possui área maior ou igual ao limiar desejado.
  3. Coloque em $\mathbf{S}$ todos os pixels $s \in R_M$ tais que $R_M(s) = 1$, inicialize l_s e k_s com o rótulo da componente conexa que contém s e o nível de referência calculado, respectivamente;
  4. Calcule $\mathcal{D}(I, \mathbf{S})$ utilizando estes parâmetros;
  5. Gere a imagem A_O, sendo $A_O(p) = D_s(p)$ para cada $p \in D_s$;

- Fechamento por área $\mathcal{A}_C(I)$: Remove da pilha de imagens binárias de I (dada pela decomposição por limiarização da imagem I) todas as componentes conexas

com valor 0 (fazendo com que elas passem a ter valor 1) cujas áreas (número de pixels pertencentes à componente) for menor do que um limiar, e gera uma imagem simplificada A_C como saída. Tal operação remove as bacias de I com área menor do que este limiar.

1. Calcule $\mathcal{R}_m(I)$ e rotule cada mínimo regional em R_m com um valor distinto;
2. Para cada mínimo regional em R_m, aplique localmente o operador preenchimento de bacias para calcular o nível de referência mais baixo em que a componente conexa que contém o máximo regional possui área maior ou igual ao limiar desejado;
3. Coloque em $\mathbf{S}$ todos os pixels $s \in R_m$ tais que $R_m(s) = 1$, inicialize l_s e k_s com o rótulo da componente conexa que contém s e o nível de referência calculado, respectivamente;
4. Calcule $\mathcal{B}(I, \mathbf{S})$ utilizando estes parâmetros;
5. Gere a imagem A_C, sendo $A_C(p) = B_s(p)$ para cada $p \in B_s$;

- Fechamento de buracos $\mathcal{C}_h(I)$: Seja I uma imagem com regiões escuras (bacias) cercadas por regiões claras. Este operador reduz o contraste entre as regiões escuras e suas vizinhanças ao aumentar os níveis de cinza das regiões escuras. A imagem simplificada resultante é chamada de C_h.

  1. Coloque em $\mathbf{S}$ todos os pixels s pertencentes à borda de I, e inicialize l_s e k_s com valor 0;
  2. Calcule $\mathcal{B}(I, \mathbf{S})$ utilizando estes parâmetros;
  3. Gere a imagem C_h, sendo $C_h(p) = B_s(p)$ para cada $p \in B_s$;

- Remoção de picos $\mathcal{R}_p(I)$: Seja I uma imagem com regiões claras (domos) cercadas por regiões escuras. Este operador reduz o contraste entre as regiões claras e suas vizinhanças ao diminuir o nível de cinza das regiões claras. A imagem simplificada resultante é chamada de R_p.

  1. Coloque em $\mathbf{S}$ todos os pixels s pertencentes à borda de I, e inicialize l_s e k_s com valor K;
  2. Calcule $\mathcal{D}(I, \mathbf{S})$ utilizando estes parâmetros;
  3. Gere a imagem R_p, sendo $R_p(p) = D_s(p)$ para cada $p \in D_s$;

Os operadores apresentados acima têm diversas aplicações na área de processamento e análise de imagens médicas [66]. Nas próximas seções, mostramos alguns operadores e o potencial de algumas combinações destes para a segmentação de imagens médicas e utilização da filtragem interativa em imagens de vídeo digital.

5.3 Aplicações em segmentação de imagens médicas

Na seção anterior mostramos que os operadores poda de domos e preenchimento de bacias representam um poderoso mecanismo para o projeto de filtros não-lineares de imagens, e que o processo de propagação de rótulos gera uma partição da imagem filtrada equivalente à que seria obtida através da transformada linhas divisoras de águas a partir de marcadores. Nesta seção mostramos três exemplos onde estes operadores conexos, juntamente com outros operadores baseados na IFT, permitem a segmentação de objetos em imagens médicas.

A Figura 5.5a apresenta uma imagem de células sangüíneas que devem ser identificadas individualmente, de maneira automática. A estratégia é primeiro separar as células do fundo da imagem. Para realizarmos esta tarefa, primeiro fazemos uma abertura por área da imagem da Figura 5.5a, obtendo uma imagem simplificada (Figura 5.5b), que é limiarizada e posteriormente suavizada através de uma abertura morfológica, resultando na Figura 5.5c. Nesta figura várias células estão se tocando, formando uma mesma componente conexa com valor 1. Para identificar cada célula individualmente, calculamos a transformada de distância Euclideana dentro destas componentes conexas, utilizando o algoritmo mostrado na seção 2.6.4, obtendo a Figura 5.5d. Os máximos regionais do mapa de distância são utilizados para identificar as células. Eles são mostrados em vermelho na Figura 5.5e, tendo sido dilatos para uma melhor visualização. Cada um destes máximos recebe um rótulo (também atribuído às suas sementes), sendo os níveis de referência inicializados em 0. Aplicamos o operador poda local de domos na imagem da Figura 5.5c. Esta operação atribui um rótulo para cada célula, sendo que os contornos em vermelho da Figura 5.5f mostram os pixels que pertencem às fronteiras das células.

A Figura 5.6a mostra uma fatia de um joelho obtida por tomografia computadorizada, onde o osso central deve ser extraído de maneira semi-automática. O usuário desenha uma linha preta fora do osso e uma linha branca dentro. Os pixels selecionados por estas linhas correspondem às sementes, cada linha correspondendo a um rótulo, e escolhemos os níveis de referência como sendo o brilho destes pixels na imagem. Uma operação de preenchimento de bacias é calculada na imagem a partir destas sementes, obtendo a imagem simplificada mostrada na Figura 5.6b. Isto define um paradigma interativo de filtragem não linear de imagens, onde regiões selecionadas têm o contraste preservado, enquanto outras são bastante simplificadas. Sabemos que a transformada linhas divisoras

de águas obtém melhores resultados quando aplicada no gradiente da imagem [3]. A Figura 5.6c mostra a segmentação obtida pelo operador preenchimento de bacias quando aplicado à imagem de gradientes da imagem representada na Figura 5.6a utilizando estas sementes, com níveis de referência dados pelos brilhos na imagem de gradientes. Podemos perceber que a transformada linhas divisoras de águas perde metade da borda do osso. Entretanto, uma família de partições pode ser obtida mudando-se o nível de referência das sementes. Podemos dilatar com um elemento estruturante a imagem de gradientes e tomar os novos brilhos como níveis de referência. A Figura 5.6d ilustra a mesma operação utilizando estes novos níveis. Podemos perceber que a segmentação melhorou, mas que ainda perde parte da borda do osso. Felizmente podemos resolver este problema utilizando nossos operadores locais, conforme descrito a seguir.

A Figura 5.7a mostra o operador fechamento de buracos aplicado à imagem da Figura 5.6a. O usuário desenha uma linha preta no interior do osso, como mostrado na Figura 5.7a, e os pixels sobre esta linha são selecionados como sementes com este rótulo. Para o operador preenchimento regional de bacias, os níveis de referência das sementes são escolhidos como sendo 1 nível acima do brilho dos pixels correspondentes, enquanto que para o operador poda regional de domos, 1 nível abaixo. O operador preenchimento regional de bacias aplicado a esta imagem permite a extração do interior do osso (Figura 5.7b). Já o operador poda regional de domos aplicado a esta mesma imagem permite a extração do osso por completo (Figura 5.7c). Além do mais, a borda do osso também pode ser isolada subtraindo-se a imagem da Figura 5.7c da imagem da Figura 5.7b, realizando um fechamento morfológico em seguida (Figure 5.7d).

Por último, apresentamos um exemplo onde a utilização de operadores conexos permite um resultado melhor do que a utilização de filtros Gaussianos na etapa de pré-processamento para a segmentação de imagens utilizando o método *live wire* [33]. A Figura 5.8a mostra o traçado obtido por este método em uma imagem de um pé obtida por ressonância magnética, onde o osso em que estamos interessados é o talus. O usuário teve que escolher sete pontos na borda para completar a segmentação. Este número é reduzido a quatro quando a imagem é simplificada por um filtro Gaussiano (Figura 5.8b). No entanto, este filtro cria bordas falsas e borra alguns detalhes importantes da fronteira. Uma filtragem melhor é obtida computando-se o *leveling* entre as imagens das Figuras 5.8a-b. A Figura 5.8c ilustra a segmentação na imagem simplificada e mostra que o número de pontos escolhidos na fronteira caiu para três, diminuindo o envolvimento do usuário no processo.

5.4 Aplicações em vídeo digital

Os operadores poda de domos e preenchimento de bacias permitem um novo paradigma de filtragem não linear e interativa, onde algumas regiões da imagem são preservadas enquanto outras são bastante simplificadas. Uma das aplicações desta abordagem é a simplificação de imagens de vídeo para transmissão em redes de baixa velocidade.

Na Figura 5.9 mostramos um exemplo de filtragem interativa. A Figura 5.9b apresenta a imagem simplificada resultante da aplicação do operador poda de domos à imagem representada na Figura 5.9a, utilizando as sementes mostradas na Figura 5.9a. Já a Figura 5.9c apresenta a simplificação da imagem da Figura 5.9b, com as sementes representadas nesta última imagem. Podemos notar que o mesa-tenista e a bolinha foram preservados nas filtragens, enquanto o fundo foi bastante simplificado. Apesar da filtragem agressiva, percebemos que na Figura 5.9b ainda podemos ter uma boa idéia da cena em que o mesa-tenista está imerso.

Gravando as imagens das Figuras 5.9a-c no formato JPEG, com a mesma qualidade de imagem (40%), são necessários, respectivamente, 8537 bytes, 5542 bytes e 2861 bytes. Desta maneira, podemos adaptar o grau de filtragem a ser aplicado de acordo com a largura de banda disponível para transmissão.

Uma vez que este método requer a especificação dos locais que não devem ser simplificados, ele se torna mais interessante quando utilizado em seqüencias onde o objeto de interesse não se movimenta muito, como por exemplo na apresentação de tele-jornais.

Um estudo mais sistemático da utilização deste paradigma de filtragem foge ao escopo deste trabalho.

5.5 Conclusão

Apresentamos neste capítulo dois operadores conexos baseados na transformada imagem-floresta: poda de domos e preenchimento de bacias. Estes operadores simplificam imagens sem a criação de bordas falsas, e produzem ao mesmo tempo uma partição da imagem simplificada que corresponde à que seria computada pela transformada linhas divisoras de águas a partir de marcadores nesta imagem. Vimos também que estes operadores podem ser aplicados localmente, constituindo uma ferramenta bastante poderosa. Mostramos como o uso destes operadores combinado com a escolha adequada de sementes permite o desenvolvimento de uma série de operadores conexos.

Foram ainda apresentados vários exemplos de aplicação destes operadores para a segmentação de imagens médicas, e uma situação onde a filtragem interativa se mostra interessante para a simplificação de vídeo digital.

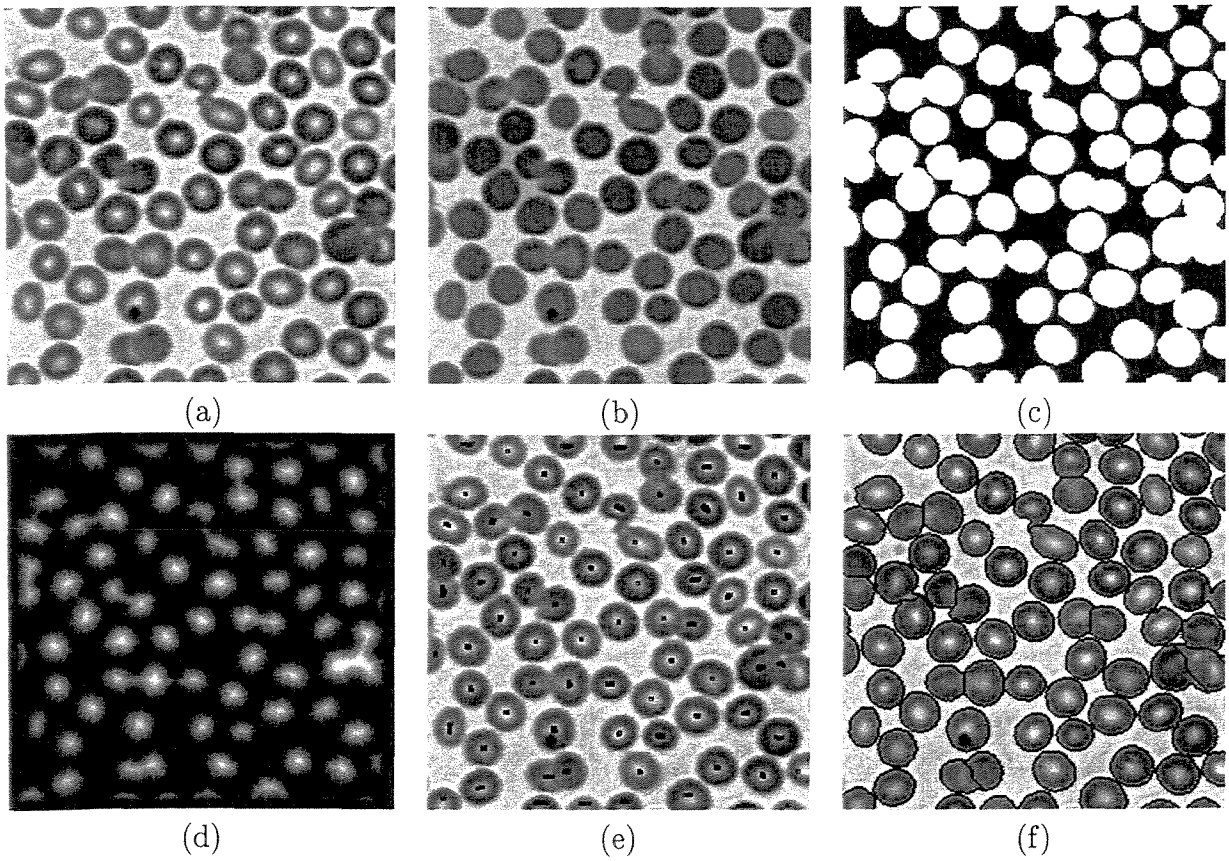


Figura 5.5: (a) Uma imagem na qual queremos identificar, de maneira automática, cada célula sangüínea. (b) Imagem (a) é simplificada por uma abertura por área. (c) Imagem (b) é limiarizada e os objetos são suavizados através de uma abertura morfológica. (d) A transformada de distância Euclideana é calculada em (c). (e) As manchas vermelhas mostram os máximos regionais de (d), após serem dilatados para possibilitar uma melhor visualização. (f) O operador poda local de domos é aplicado em (c), onde cada célula recebe um rótulo que serve para distingui-las

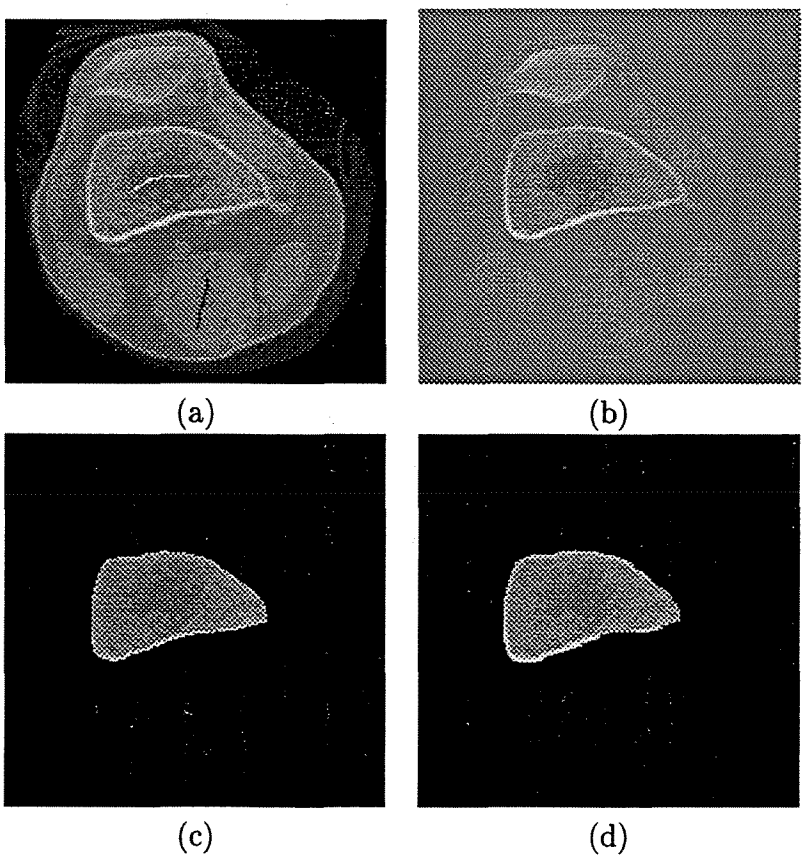


Figura 5.6: (a) Uma fatia de um joelho obtida através de tomografia computadorizada, onde as linhas branca e preta indicam as sementes dentro e fora do objeto de interesse, no caso o osso ao centro, respectivamente. (b) A operação de preenchimento de bacias é computada em (a) utilizando estas sementes. A mesma operação é calculada na imagens de gradientes de (a), e na dilatação da imagem de gradientes de (a) por um elemento estruturante. As partições obtidas são mostradas em (c) e (d), respectivamente.

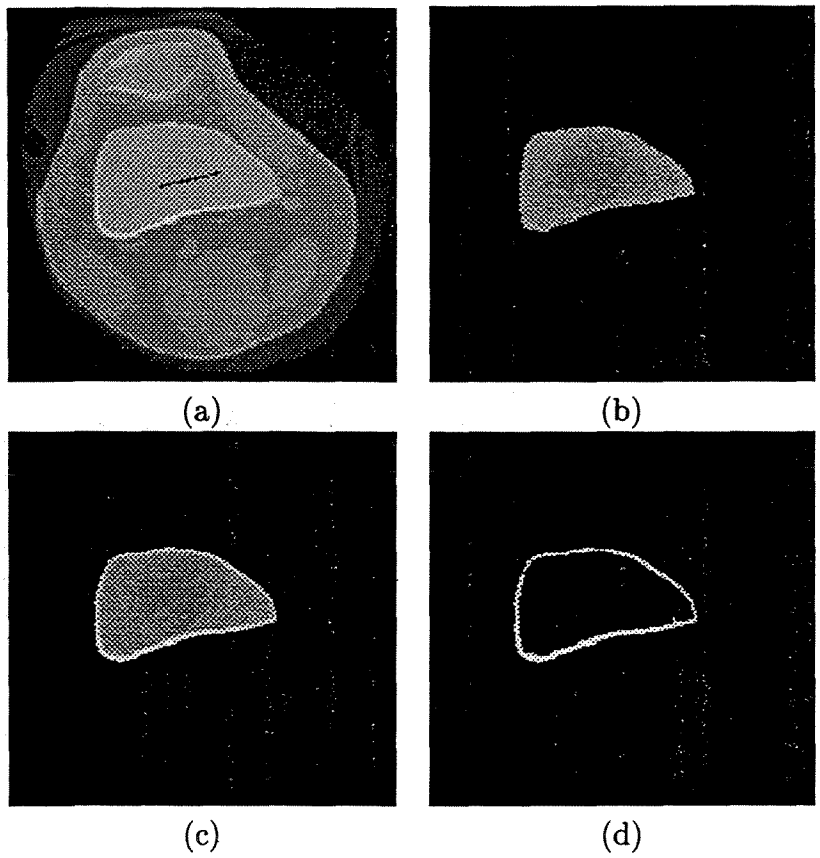
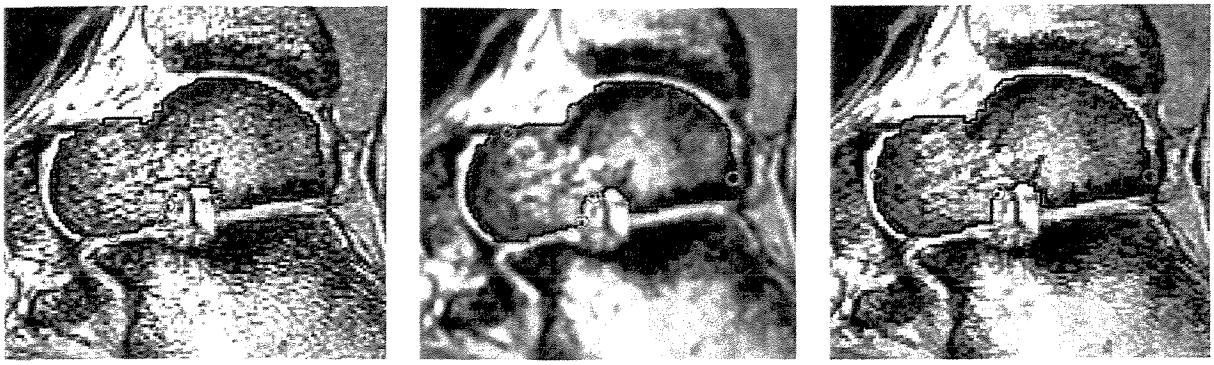


Figura 5.7: (a) Operador fechamento de buracos aplicado à imagem da Figura 5.6a. Sementes são desenhadas dentro do osso, e os operadores preenchimento regional de bacias e poda regional de domos são aplicados para extrair o interior do osso (b) e o osso por completo (c). (d) a borda do osso é o resultado de se fazer (c) menos (b), seguido de um fechamento morfológico.



(a) (b) (c)

Figura 5.8: Esta figura ilustra as melhorias obtidas na segmentação através do método *live wire* ao se fazer uma suavização da imagem em uma etapa de pré-processamento. (a) traçado necessário para se segmentar o osso talus sem a pré-filtragem. (b) resultado obtido aplicando-se um filtro Gaussiano a (a). (c) resultado ainda melhor, utilizando-se a imagem simplificada obtida com a aplicação do operador conexo *leveling* entre as imagens (a) e (b).

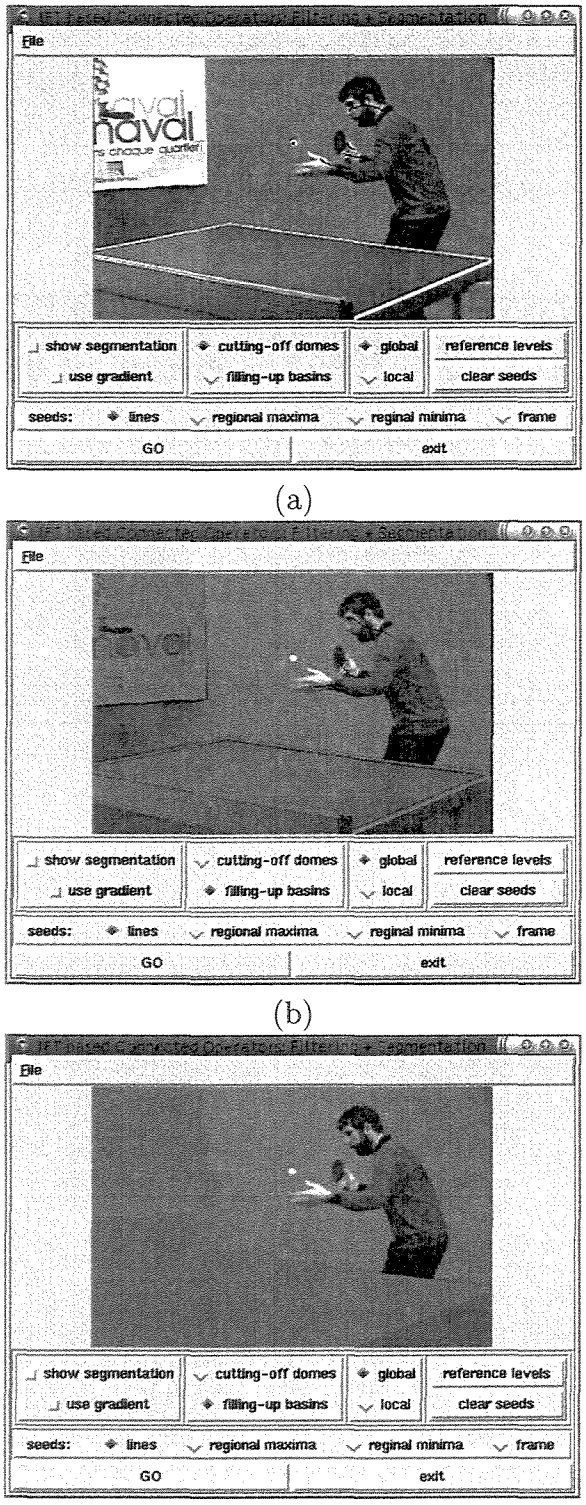


Figura 5.9: Filtragem interativa: (a) imagem original, (b) poda de domos de (a) a partir das sementes mostradas em (a), (c) preenchimento de bacias de (b) com sementes mostradas em (b).

Capítulo 6

Conclusão e sugestões de trabalhos futuros

Neste trabalho exploramos o desenvolvimento de diversos operadores de processamento e análise de imagens através da transformada imagem-floresta. Os resultados obtidos foram interessantes, possibilitando a publicação de três artigos em congressos e revistas no exterior [29, 30, 31].

Fizemos uma introdução à transformada imagem-floresta, fazendo uma apresentação de como os problemas de processamento e análise de imagens devem ser modelados para serem resolvidos através da IFT, as condições em que a IFT gera uma partição ótima, o algoritmo básico, uma prova original de sua corretude e uma análise de sua complexidade. Mostramos ainda como transformadas de distância utilizando diversas métricas podem ser implementadas através da IFT, com uma nova análise de complexidade.

Mostramos que diversos métodos clássicos de segmentação podem ser facilmente implementados através da IFT. A formulação da transformada linhas divisoras de água através da IFT já havia permitido um melhor entendimento das condições de otimalidade da LDA. Neste trabalho acrescentamos os métodos baseados em conexidade *fuzzy*, que têm sido aplicados com muito sucesso em diversos trabalhos descritos na literatura.

O método de geração de esqueletos multi-escala através da IFT permite a geração de esqueletos baseados na métrica Euclideana e conexos em todas as escalas, podendo ser utilizado em objetos com topologia arbitrária. Ele permite ainda a geração dos esqueletos por zona de influência entre objetos. Os exemplos que foram mostrados atestam a alta qualidade dos esqueletos gerados, comparável aos melhores algoritmos descritos na literatura, com a vantagem do método apresentado ser bastante simples e eficiente. Todas estas características possibilitam que ele seja utilizado em importantes aplicações, abrindo novas possibilidades de pesquisa como o *data mining* para neuromorfometria. Apresentamos ainda uma prova original de que, pelo menos no caso contínuo, os esqueletos calculados

são realmente conexos em todas as escalas.

Por fim foram apresentados os operadores conexos poda de domos e preenchimento de bacias, em suas versões globais e locais. Estes operadores possibilitam a simplificação de imagens sem a criação de bordas falsas, e a geração simultânea de uma partição da imagem simplificada equivalente à obtida pela transformada linhas divisoras de águas. Mostramos ainda como implementar diversos operadores conexos a partir destes operadores básicos, além de exemplos de utilização em imagens médicas e vídeo digital.

Finalmente, sugerimos as seguintes linhas de pesquisa para trabalhos futuros:

- exploração de técnicas de pré-segmentação: enquanto seres humanos vêem as imagens em termos globais, os computadores atualmente têm operado basicamente em níveis locais, atuando diretamente sobre os pixels. Pode ser interessante a geração de mosaicos que agrupem em uma mesma região pixels que são bastante similares. Os mosaicos podem ser gerados através de diversos métodos, como limiarizações de árvores geradoras mínimas, análise das zonas planas da imagem [62], métricas baseadas em volume [68, 73], linhas divisoras de águas a partir de mínimos regionais, crescimento de regiões por similaridade ou técnicas de *clustering*. A partir deste mosaico, podemos trabalhar em uma representação intermediária para imagens, baseada em grafos de regiões. Uma aplicação destas técnicas poderia ser uma tentativa de melhoria da atribuição de custos às arestas, onde pixels dentro de uma mesma região gerariam arestas com custos baixos, e pixels em regiões distintas arestas com custos mais altos;

- exploração de técnicas de treinamento para a melhoria das funções que atribuem pesos às arestas;

- exploração mais sistemática do potencial de segmentação associado aos operadores conexos, em especial nas versões regionais, e combinação de operadores para obter melhores segmentações;

- elaboração de descritores que utilizem os esqueletos multi-escala;

- criação de novos operadores que possam utilizar a IFT, como operadores de cálculo de cascos convexos.

Apêndice A

A fila hierárquica de prioridades

Conforme foi dito no Capítulo 2, quando utilizamos funções suaves de custo de caminho que assumem apenas valores inteiros, e a diferença entre o valor desta função entre quaisquer dois vértices consecutivos de um caminho é um inteiro entre 0 e C, podemos utilizar uma fila hierárquica de prioridades.

Esta estrutura, representada na Figura A.1, consiste de uma fila circular com $C+1$ *buckets* numerados de 0 a C, cada *bucket* i_0 contendo uma lista duplamente ligada e circular que representa todos os vértices que possuem o mesmo custo de caminho a partir das sementes, em um determinado instante. A posição na fila circular em que um vértice se encontra pode ser dada pelo resto da divisão inteira de X por $C+1$, sendo X o custo de caminho até este vértice no instante considerado. Processamos seqüencialmente as listas de cada posição da fila circular, de maneira a satisfazer as prioridades associadas aos vértices contidos nas listas. Estas listas duplamente ligadas são estruturas FIFO que permitem inserções e remoções em tempo constante, uma vez que inserções são realizadas no final das listas e remoções no começo – uma conseqüência do fato da função de custo de caminho assumir apenas valores inteiros.

Para mostrar que a limitação no peso das arestas no valor C é necessária, vamos supor que estamos processando vértices cujos custos de caminho sejam X, com o resto da divisão inteira de X por $C+1$ igual a i_0, ou seja, os vértices da posição i_0 da fila circular. Suponha que a seja um destes vértices, com a aresta que liga a a um vértice b tendo peso $C+1$. Neste caso, inserimos b na posição $X+C+1$ resto $C+1$, que é igual a X resto $C+1$, ou ainda i_0. Assim, iremos processar b antes de processar outros vértices que já estão na estrutura e que possuem custo de caminho menor do que o custo até b, violando a prioridade destes vértices.

Para que esta estrutura seja ainda mais eficiente, é importante que evitemos pedidos de alocação de memória dinâmica para os nós destas listas. Isto pode ser feito utilizando-se um vetor de dimensão igual ao número de pixels da imagem, contendo em cada posição

um índice para o vértice predecessor e outro para o sucessor na lista. A única limitação imposta por esta abordagem é que um determinado vértice só pode estar em uma posição na fila por vez, o que é bem razoável tendo em vista o significado associado a esta posição.

A utilização desta estrutura permite a diversos operadores baseados na transformada imagem-floresta uma execução em tempo linear em relação ao tamanho da imagem.

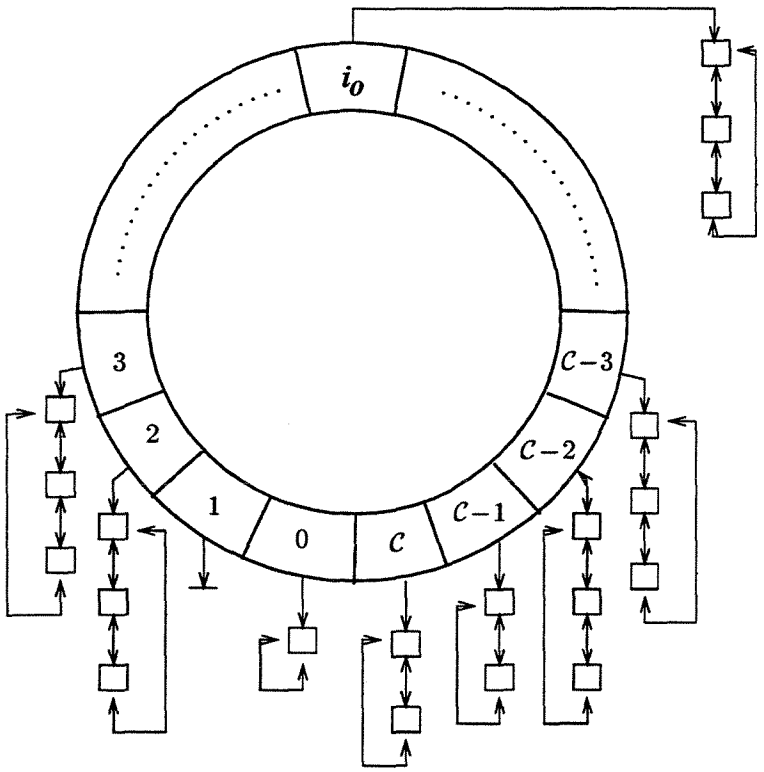


Figura A.1: Estrutura fila hierárquica de prioridades. Temos uma fila circular, onde cada posição tem associada uma lista duplamente ligada e circular que contém todos os vértices com um mesmo custo de caminho a partir das sementes, em um determinado instante.

Apêndice B

Interfaces implementadas

Foram desenvolvidas duas interfaces em linguagem Tcl/Tk, para as ferramentas de esqueletos multi-escala (capítulo 4) e para os operadores conexos (capítulo 5). Estas ferramentas estarão disponibilizadas na URL http://www.ic.unicamp.br/~afalcao, para os sistemas Gnu/Linux e Solaris.

B.1 Esqueletos multi-escala

A interface para a utilização de esqueletos multi-escala é de fácil utilização. Para executá-la, digitamos:

wish skel.tcl <imagem.pgm >

São abertas três janelas. A primeira, mostrada na Figura B.1 corresponde à tela principal, onde aparecem a imagem de entrada e uma barra de rolagem que permite a escolha da escala em que os esqueletos devem ser representados. As outras duas janelas correspondem aos esqueletos (Figura B.2a) e à forma filtrada (Figura B.2b), na escala determinada na janela principal. Sempre que a escala é alterada na janela principal, as outras duas são atualizadas para refletir a mudança. As Figuras B.3a e B.3b mostram o esqueleto e a forma filtrada em uma outra escala.

Na janela principal temos também uma barra de menus, com duas opções principais: *file* e *options*. A opção *file* permite que outra imagem seja utilizada e que se saia da ferramenta. A opção *options* permite que escolhamos que tipo de esqueletos e SKIZ queremos visualizar, dentre as opções interno, externo e completo, conforme descrito no capítulo 4.

Figura B.1: Tela principal da ferramenta de geração de esqueletos multi-escala, mostrando a imagem de entrada e a barra que permite escolher a escala em que os esqueletos são representados.

B.2 Operadores Conexos

A interface para a utilização dos operadores conexos baseados na IFT é de utilização mais complexa. Para executá-la, digitamos:

wish filter.tcl <imagem.pgm >

Inicialmente, é mostrada a tela principal da ferramenta, como ilustrado na Figura B.4. Nesta tela são mostradas a imagem a ser simplificada, e as diversas opções que podem ser escolhidas.

O botão *reference levels* abre uma outra janela, que permite que escolhamos os níveis de referência das sementes através de uma série de opções de pré-filtragem. Descreveremos esta janela em detalhes mais adiante.

São oferecidos quatro tipos de sementes: *lines*, *regional maxima*, *regional minima* e

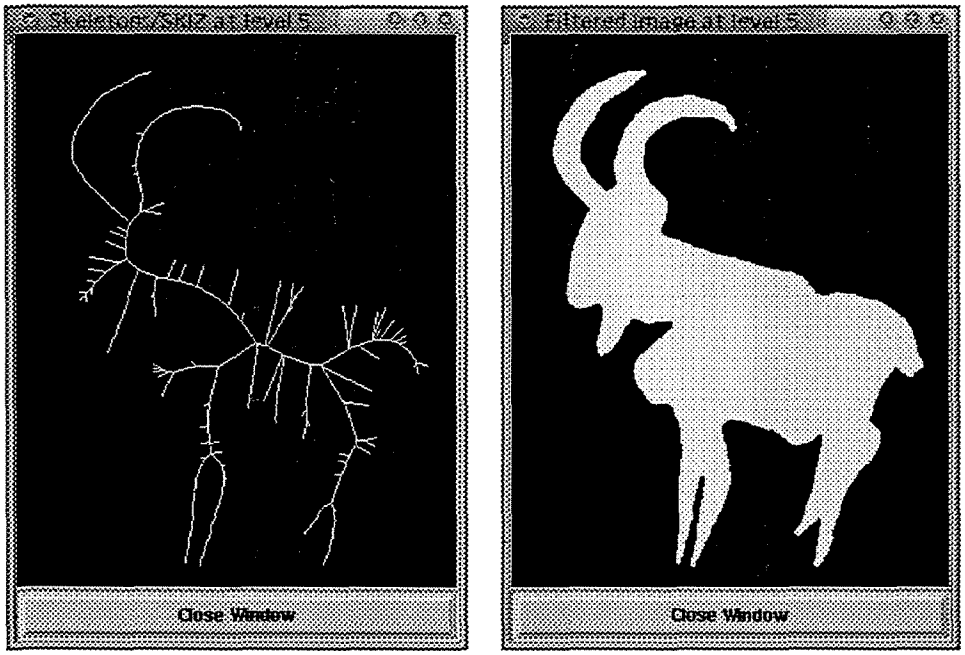


Figura B.2: Esqueleto e forma filtrada da imagem da Figura B.1, na escala 5.

frame. A opção *lines* permite que escolhamos manualmente as sementes, desenhando-as na imagem de entrada. Ao clicarmos o mouse e o arrastarmos pela imagem escolhemos as sementes. Todas as sementes que forem escolhidas com um mesmo botão do mouse apertado possuirão o mesmo rótulo. As opções *regional maxima* e *regional minima* fazem com que utilizemos os máximos e os mínimos regionais da imagem, calculados na imagem que estabelece os níveis de referência das sementes. Cada máximo ou mínimo regional receberá um rótulo diferente. Já a opção *frame* faz com que as bordas da imagem sejam as sementes, permitindo, por exemplo a realização das operações fechamento de buracos e remoção de picos.

O botão *clear seeds* limpa as sementes que foram desenhadas pelo usuário, no modo de escolha de sementes *lines*.

O botão *show segmentation*, quando ativado, faz com que a segmentação obtida seja superposta na imagem de entrada, mostrando quais estruturas desta foram detectadas. Se as sementes tiverem um mesmo rótulo nenhuma segmentação será mostrada.

O botão *use gradient* permite com que utilizamos não a imagem de entrada na operação conexa, mas o gradiente da mesma.

Os botões *cutting-off domes* e *filling-up basins* permitem a realização das operações poda de domos e preenchimento de bacias, respectivamente.

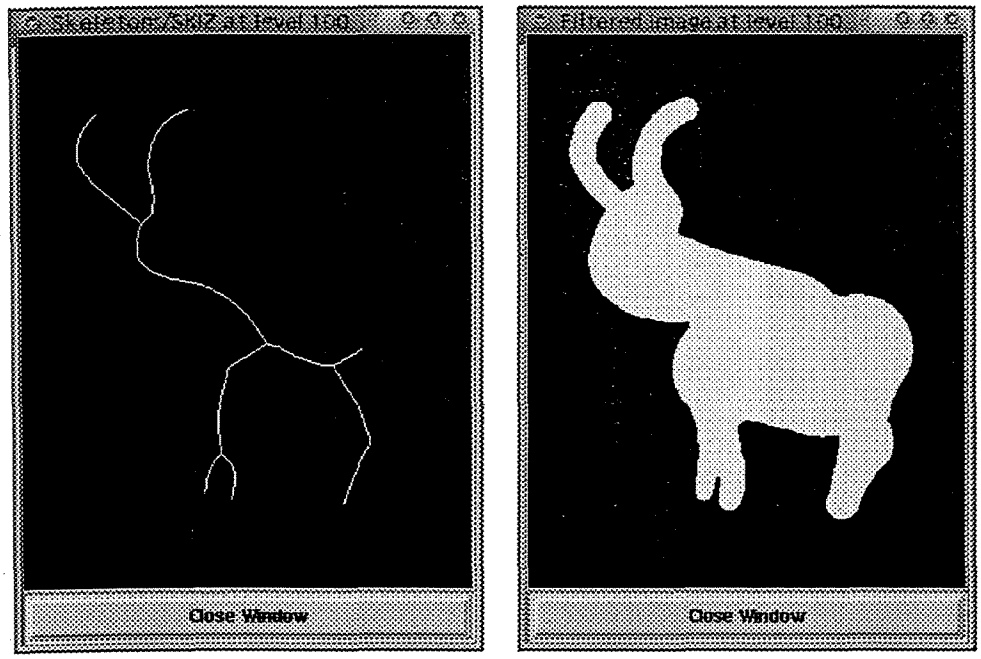


Figura B.3: Esqueleto e forma filtrada da imagem da Figura B.1, na escala 100.

Os botões *global* e *local* permitem que escolhamos o escopo em que os operadores poda de domos e preenchimento de bacias devem atuar, global ou local, respectivamente, permitindo as versões poda local de domos e preenchimento local de bacias, conforme descrito no capítulo 5.

O botão *Go* faz com que a operação conexa selecionada seja executada.

Por fim, o botão *exit* fecha a ferramenta.

Existe ainda um menu *file*, que permite que abramos novas imagens, salvemos o resultado da operação ou saiamos da ferramenta. Podemos abrir outra imagem a ser simplificada ou uma imagem que será utilizada para a determinação dos níveis de referência das sementes, permitindo a utilização de ferramentas de pré-processamento externas para esta tarefa. Podemos salvar duas imagens: a imagem simplificada resultante da operação conexa e a imagem de rótulos correspondente à segmentação obtida.

Quando pressionamos o botão *reference levels*, é aberta uma nova janela, mostrada na Figura B.5. Nesta janela, temos mostrada a imagem que estabelece os níveis de referência para as sementes, e uma série de opções de pré-processamento. Quando a ferramenta é aberta, a imagem que aparece nesta janela corresponde à imagem de entrada. Podemos entrar com um valor para uma opção de pré-processamento. Quando pressionamos *view*, a última opção de pré-processamento escolhida é executada, e o resultado é mostrado nesta

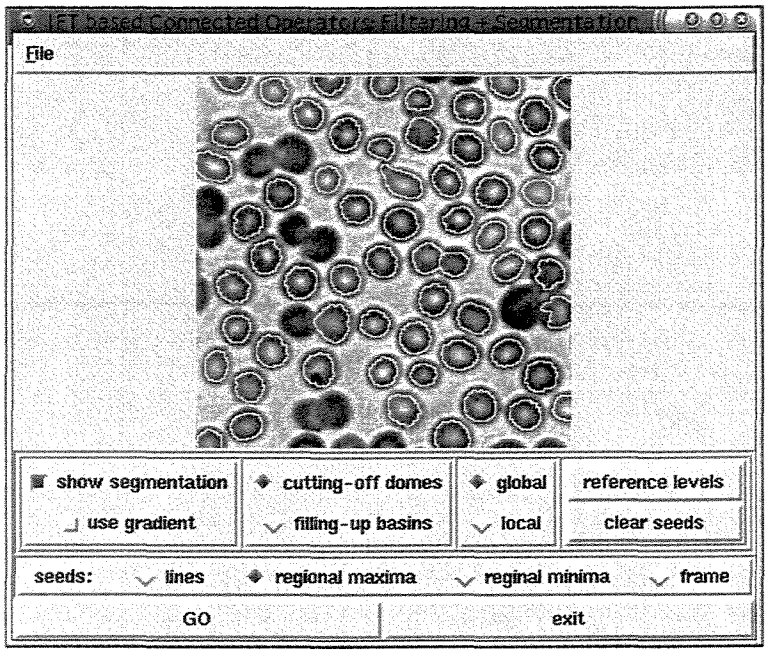


Figura B.4: Tela principal da ferramenta operadores conexos.

janela. Para aceitar tal operação, devemos pressionar o botão *accept*. Se este botão não for pressionado, o resultado é desconsiderado. Podemos cancelar a última operação realizada, desde que não tenhamos pressionado o botão *accept*, ao pressionar o botão *cancel*, que também fecha esta janela. Quando pressionamos *accept*, o botão *cancel* passa a se chamar *close*, indicando que a operação não poderá ser desfeita, servindo o botão apenas para o fechamento da janela. Diversas operações de pré-processamento podem ser realizadas para a escolha dos níveis de referência, desde que o botão *accept* seja pressionado entre cada operação.

As opções de pré-processamento permitidas são as seguinte:

- *erode*: realiza a operação de erosão morfológica, com elemento estruturante circular, de raio igual à raiz quadrada do valor entrado;
- *dilate*: realiza a operação de dilatação morfológica, com elemento estruturante circular, de raio igual à raiz quadrada do valor entrado;
- *close*: realiza a operação de fechamento morfológico, com elemento estruturante circular, de raio igual à raiz quadrada do valor entrado;
- *open*: realiza a operação de abertura morfológica, com elemento estruturante circular, de raio igual à raiz quadrada do valor entrado;

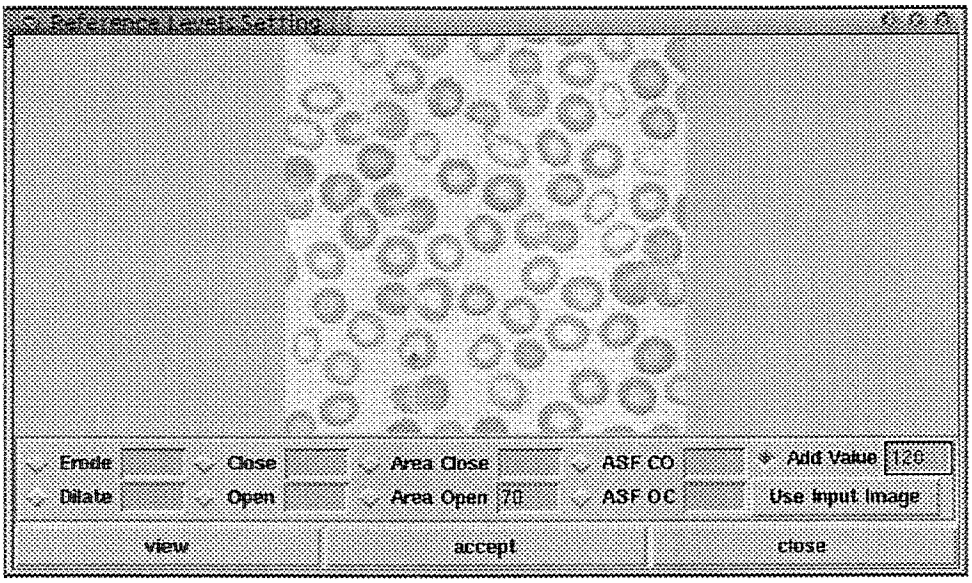


Figura B.5: Tela que permite a escolha do nível de referência das sementes a partir de diversas operações de pré-processamento.

- *area close*: realiza a operação fechamento por área, com valor de área igual ao valor entrado;
- *area open*: realiza a operação abertura por área, com valor de área igual ao valor entrado;
- *ASF CO*: realiza filtragem alternada seqüencial do tipo fechamento seguido de abertura, com elemento estruturante circular, de raio igual à raiz quadrada do valor entrado;
- *ASF CO*: realiza filtragem alternada seqüencial do tipo abertura seguida de fechamento, com elemento estruturante circular, de raio igual à raiz quadrada do valor entrado;
- *add value*: adiciona a cada pixel da imagem de níveis de referência corrente o valor entrado, podendo ser um número positivo ou negativo. Se for um número positivo, o valor resultante da soma é limitado a 255. Se for negativo, o menor valor obtido é zero.

A imagem simplificada resultante da atuação dos filtros conexos é apresentada em outra janela, conforme mostrado na Figura B.6.

A segmentação mostrada na Figura B.4 foi obtida seguindo-se os seguintes passos: escolhemos as opções *show segmentation*, *cutting-off domes*, *global*, *regional maxima*. Em seguida, abrimos a janela de escolha dos níveis de referência das sementes, pressionando o botão *reference levels*. Nesta janela, fizemos uma abertura por área de valor 70 e pressionamos *accept*. Ainda nesta janela, realizamos a operação *add value* com valor 120. Novamente pressionamos *accept*. A imagem com os níveis de referência finais está mostrada na Figura B.5. Ao pressionarmos *Go* na janela principal, obtemos a segmentação mostrada na janela principal, e a imagem simplificada, mostrada na Figura B.6. Esta segmentação corresponde à escolha das células que possuem núcleos com área maior que 70 e que não estejam unidos ao fundo da imagem.

Já na Figura B.7 apresentamos a segmentação obtida utilizando os mesmos parâmetros do exemplo da Figura B.4, mas utilizando o operador poda regional de domos. Conseguimos identificar as células e o núcleos com área maior que 70 pixels, apesar de algumas células não terem sido separadas. Combinando a informação dos núcleos da Figura B.7 com a informação das células da Figura B.4, conseguimos extrair cada célula que possui núcleo com área maior que 70 pixels, juntamente com seus núcleos, como mostrado na Figura B.8.

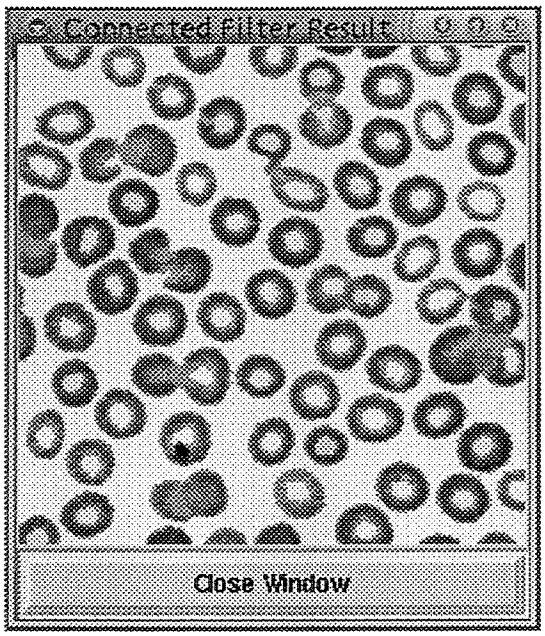


Figura B.6: Tela que mostra a imagem simplificada resultante da operação conexa.

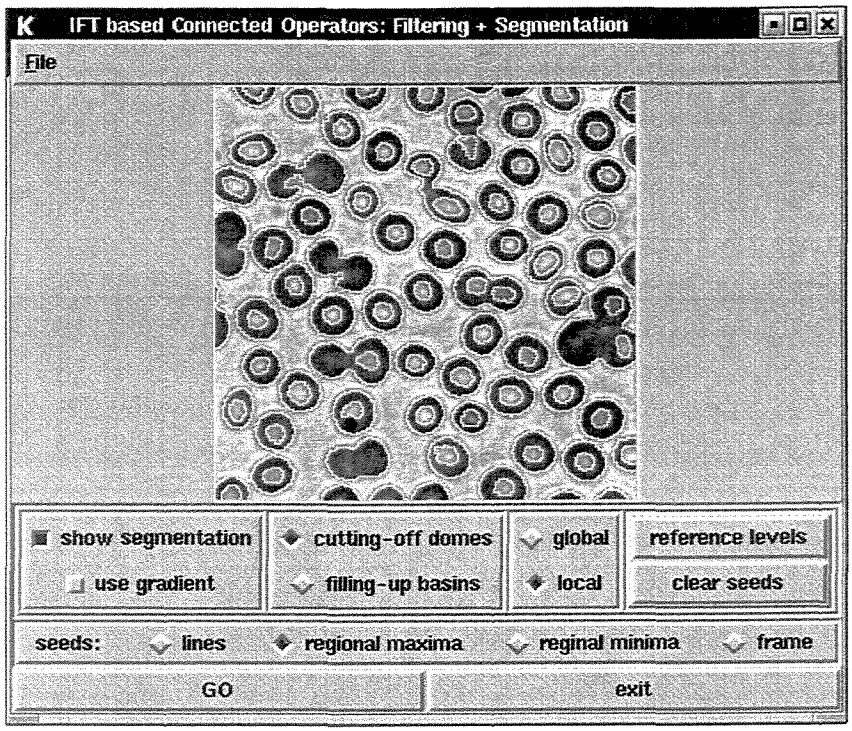


Figura B.7: Segmentação obtida utilizando o operador poda regional de domos.

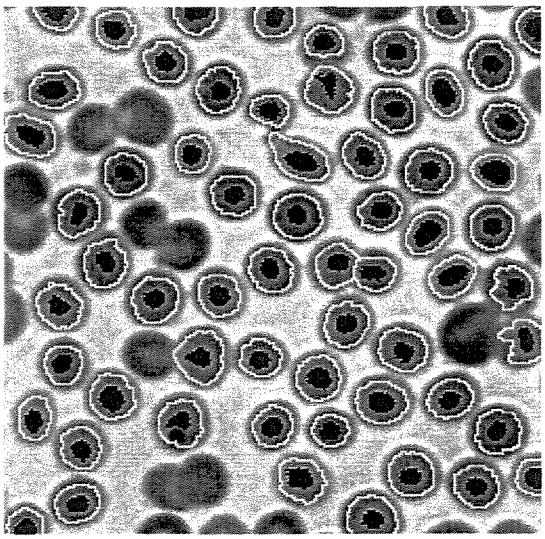


Figura B.8: Imagem que mostra a combinação das células identificadas na Figura B.4 com os núcleos identificados na Figura B.7.

Bibliografia

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[2] J. Barrera, G. Banon, R. de A. Lotufo, and R. Hirata Jr. Mmach: A mathematical morphology toolbox for the khoros system. *Journal of Electronic Imaging*, 7(1):174–210, Jan 1998.

[3] 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, 1993.

[4] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form.* MIT Press, Cambridge, MA, 1967.

[5] B. Bollobás. *Graph Theory: An Introductory Course.* Springer Verlag, 1979.

[6] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications.* MacMillan, London, 1976.

[7] M. Brady. Criteria for representations of shape. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 39–84. Academic Press, New York, 1983.

[8] J.W. Brandt and V.R. Algazi. Continuous skeleton computation by voronoi diagram. *Computer Vision, Graphics and Image Processing*, 55(3):329–338, 1992.

[9] J. Canny and B. Donald. Simplified voronoi diagrams, A.I. Memo 957. Technical report, MIT, Boston, MA, 1987.

[10] J. Chassery and M. Melkemi. A segmentation method using voronoi diagrams in a split and merge environment. In *Progress in Image Analysis and Processing, Proc. 5th Int. Conf. on Image Analysis and Processing, 1989, Positano, Italy*, pages 44–48, Singapore, 1990. World Scientific.

[11] B. Cherkassky, A. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. In *Proceedings, 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 516–525, 1994.

[12] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and ballons for 2-d and 3-d images. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 15(11), Nov 1993.

[13] Cormen, Leiserson, and Rivert. *Introduction to Algorithms.* McGraw Hill Book Company, 1990.

[14] O. Cuisenaire. *Distance Transformations: fast algorithms and applications to medical image processing.* PhD thesis, Université Catholique de Louvain, Bélgica, 1999.

[15] O. Cuisenaire and B. Macq. Fast Euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, 76(1):163–172, Nov 1999.

[16] L. da F. Costa, A. G. Campos, L. F. Estrozi, L. G. Rios-Filho, and A. Bosco. A biologically-motivated approach to image representation and its application to neuromorphometry. In *Lecture Notes in Computer Science*, volume 1811, pages 407–416, Seoul, Korea, May 2000.

[17] L. da F. Costa and L. F. Estrozi. Multiresolution shape representation without border shifting. *Electronics Letters*, 35(21):1829–1830, Oct 1999.

[18] L. da F. Costa and R. M. Cesar Jr. *Shape Analysis and Classification: Theory and Practice.* CRC Press, Boca Raton, 2000.

[19] L. da F. Costa, R. M. Cesar Jr., and R. C. Coelho. *Analysis and Synthesis of Morphologically Realistic Neural Networks*, chapter 18, pages 505–528. Harwood Academic Publishers, 1999.

[20] L. da F. Costa and R. C. Coelho R. M. Cesar Jr. Computer vision based morphometric characterization of neural cells. *Review of Scientific Instruments*, 66(7):3770–3773, 1995.

[21] L. da F. Costa and T. Velte. Automatic characterization and classification of ganglion cells from the salamander retina. *Journal of Comparative Neurology*, 404(1):33–51, 1999.

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

[23] E. de A. Rodrigues. Segmentação de imagens tomográficas usando contornos deformáveis com segmentos de custo mínimo. Master's thesis, Universidade Estadual de Campinas, 2000.

[24] F de A. Zampirolli. Operadores morfológicos baseados em grafos de vizinhança. Master's thesis, Universidade de São Paulo, 1997.

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

[26] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[27] A.R. Dill, M.D. Levine, and P.B. Noble. Multiple resolution skeletons. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 9:485–504, 1987.

[28] R. Duda and P. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, 1973.

[29] A.X. Falcão and B. S. da Cunha. Multiscale shape representation by image foresting transform. In *Proceedings of SPIE on Medical Imaging'2001*, pages 1091–1100, San Diego, CA, Feb 2001.

[30] A.X. Falcão, B. S. da Cunha, and R. A. Lotufo. Design of connected operators using the image foresting transform. In *Proceedings of SPIE on Medical Imaging'2001*, pages 468–479, San Diego, CA, Feb 2001.

[31] A.X. Falcão, L. da F. Costa, and B.S. da Cunha. Multiscale Skeletons by Image Foresting Transform and its application to neuromorphometry. *Pattern Recognition*, 2001. accepted for publication.

[32] A.X. Falcão, J. Stolfi, R. A. Lotufo, B.S. da Cunha, and G. Araujo. The image foresting transform. *Computer Vision and Image Understanding*, 2001. in preparation.

[33] A.X. Falcão, J.K. Udupa, and F.K. Miyazawa. An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly. *IEEE Transactions on Medical Imaging*, 19(1):55–62, Jan 2000.

[34] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, and R.A. Lotufo. User-steered image segmentation paradigms: Live-wire and live-lane. *Graphical Models and Image Processing*, 60(4):233–260, Jul 1998.

[35] R. Gonzales and R. Woods. *Digital Image Processing.* Addison Wesley, 1992.

[36] T. Grigorishin and Y.H. Yang. Skeletonization: An Eletrostatic Field-Based Approach. *Pattern Analysis and Applications*, 1:163–177, 1998.

[37] M. Grimaud. A new measure of contrast: the dynamics. In *Proceedings SPIE Visual Communications and Image Processing'92*, pages 292–305, San Diego, CA, Jul 1992.

[38] Z. Guo and R.W. Hall. Fast fully parallel thinning algorithms. *Computer Vision, Graphics and Image Processing*, 55(3):317–328, 1992.

[39] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29(1):100–132, Jan 1985.

[40] M. Ilg. Knowledge-based interpretation of road maps. In *Proceedings 4th International Symposium on Spatial Data Handling*, pages 25–34, Zürich, 1990.

[41] R. M. Cesar Jr. and L. da F. Costa. Application and assessment of multiscale bending energy for morphometric characterization of neural cells. *Review of Scientific Instruments*, 68(5):2177–2186, 1997.

[42] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1:321–331, 1987.

[43] R. Kimmel, A. Amir, and A.M. Bruckstein. Finding shortest paths on surfaces using level sets propagation. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 17(1):635–640, 1995.

[44] F. Klein and O. Kubler. Euclidean distance transformation and model guided image interpretation. *Pattern Recognition Letters*, 3:19–30, 1987.

[45] L. Lam, S.E. Lee, and C.Y. Suen. Thinning methodologies - a comprehensive survey. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 14(9):869–885, 1992.

[46] Ch. Lantuéjoul and S. Beucher. On the use of geodesic metric in image analysis. *Journal of Microscopy*, 121:39–49, 1981.

[47] F. Leymarie and M.D. Levine. Simulating the grassfire transform using an active contour model. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 14(1):56–75, 1992.

[48] R.A. Lotufo and A.X. Falcão. The ordered queue and the optimality of the watershed approaches. In *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 341–350. Kluwer Academic Publishers, Palo Alto, USA, Jun 2000.

[49] R.A. Lotufo, A.X. Falcão, and F.A. Zampirolli. Fast Euclidean distance transform using a graph-search algorithm. In *XIII Brazilian Symposium on Computer Graphics and Image Processing*, pages 269–275, Gramado - RS, Brazil, Oct 2000.

[50] F. Meyer. Minimum spanning forests for morphological segmentation. In J. Serra and P. Soille, editors, *Mathematical Morphology and Its Applications to Image Processing*, pages 77–84. Kluwer Academic Publishers, 1994.

[51] F. Meyer. The levelings. In H. Heijmans and J. Roerdink, editors, *4th International Symposium on Mathematical Morphology*, pages 190–207. Kluwer Academic, 1998.

[52] U. Montanari. Continuous skeletons from digitezed images. *J. Assoc. Comput. Machinery*, 16(4):534–549, 1969.

[53] E. Moore. The shortest path throught a maze. In *Proc. Internat. Symposium on The Theory of Switching, Part II*, Cambridge, MA, 1957. Harvard University Press.

[54] E.N. Mortensen and W.A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60:349–384, 1998.

[55] R. L. Ogniewicz and O. Kübler. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28(3):343–359, 1996.

[56] R.L. Ogniewicz and M. Ilg. Voronoi skeletons: theory and applications. In *Proceedings Conf. on Comp. Vision and Pattern Recognition*, pages 63–69, Illinois, Jun 1992.

[57] F.P. Preparata and M.I. Shamos. *Computational Geometry, Texts and Monographs in Computer Science.* Springer-Verlag, New York, second edition, 1990.

[58] A. Rosenfeld. Axial representations of shape. *Computer Vision, Graphics and Image Processing*, 33:156–173, 1986.

[59] P.K. Saha and J. Udupa. Relative fuzzy connectedness among multiple objects: theory, algorithms and applications in image segmentation. *Computer Vision and Image Understanding*, 82:1–15, 2001.

[60] P. Salembier and A. Oliveras. Practical extensions of conennected operators. In P. Maragos, R. W. Schafer, and M. A. Butt, editors, *Proc. Int. Symp. Mathematical Morphology*, pages 97–110, May 1996.

[61] 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.

[62] P. Salembier and J. Serra. Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Transactions on Image Processing*, 4(8):1153–1160, Aug 1995.

[63] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of IEEE Conf. on Comp. Vision and Pattern Recognition*, Puerto Rico, Jun 1997.

[64] R. W. Smith. Computer processing of line images: A survey. *Pattern Recognition*, 20(1):7–15, 1987.

[65] P. Soille. *Morphological Image Analysis: Principles and Applications.* Springer, 1999.

[66] M. Sonka and J.M. Fitzpatrick, editors. *Handbook of Medical Imaging: Volume 2. Medical Image Processing and Analysis - Chapter 3.* SPIE Press, USA, 2000.

[67] J. Udupa and S. Samarasekera. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Computer Vision, Graphics and Image Processing*, 58(3):246–261, May 1996.

[68] C. Vachier. *Extraction de caractéristiques, segmentation d'image et morphologie mathématique.* PhD thesis, École des Mines de Paris, França, 1995.

[69] L. Vincent. Morphological area opening and closings for greyscale images. In *Shape in Picture'92 - NATO Workshop*, Driebergen, The Netherlands, Sep 1992. Springer Verlag.

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

[71] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* Wiley, New York, second edition, 1982.

[72] Y. Xia. Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(10):1076–1086, 1989.

[73] F. Zanoguera, B. Marcotegui, and F. Meyer. A toolbox for interactive segmentation based on nested partitions. In *ICIP'99 : International Conference on Image Processing.* Oct 1999.

Índice