Universidade Federal Fluminense

GIULIANO PRADO DE MORAIS GIGLIO

Detecção de bordas usando a Transformada Imagem-Floresta

NITERÓI

2005

Universidade Federal Fluminense

GIULIANO PRADO DE MORAIS GIGLIO

Detecção de bordas usando a Transformada Imagem-Floresta

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Orientador:
Helena Cristina da Gama Leitão

NITERÓI

2005

Detecção de bordas usando a Transformada Imagem-Floresta

Giuliano Prado de Morais Giglio

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada em novembro de 2005 por:

Profa. Dra. Helena Cristina da Gama Leitão, D. Sc. - Orientadora
IC-UFF

Prof. Dr. Alexandre Falcão, Ph.D.
IC-UNICAMP

Prof. Dr. José Ricardo de Almeida Torreão, Ph.D.
IC-UFF

Niterói
2005.

Dedico esta vitória a minha querida mãe Zélia, minha filha Giulia,
aos meus avós Lourdes e José Henrique (in memoriam)
e a Deus que torna tudo capaz àquele que, com fé e esperança, crê.

Agradecimentos

A Deus, que me deu a vida e a graça de viver este momento. *"Porque dEle e por meio dEle e para Ele são todas as coisas"* (Rm 11.36).

À Nossa Senhora de Aparecida por quem minhas orações em momentos de pressão, angústia, dúvidas foram amorozamente acolhidas, em sua interseção, me encaminhado às vitórias. *"Por isto, desde agora, me proclamarão bem-aventurada todas as gerações, porque realizou em mim maravilhas aquele que é poderoso e cujo nome é santo."* (Luc 01,48-49).

À minha mãe Zélia pelo seu amor, solidariedade, confiança, ternura e incentivo. Agradeço seus esforços incondicionais e o exemplo de luta pela vida. Ao meu pai Ricardo, que nunca deixou de "apostar" na minha capacidade e pelos grandes incentivos.

Às mulheres da minha vida, minha esposa Lilian e minha querida filha Giulia, pois não teria conseguido sair um busca desse sonho se não fosse elas terem-no sonhado junto comigo. Obrigado por não me deixarem desistir em momentos que parecia tão difícil continuar!

À minha querida irmã Cristiane ... nossa amizade fraterna será sempre um dos nossos alicerces e contar com suas palavras, incentivos, conselhos me fazem seguir em frente com toda a força que Deus me dá através de ti.

À professora Helena Cristina da Gama Leitão, pela sua orientação, sempre estando presente para esclarecer dúvidas, incentivando cada passo dos meus estudos e sempre me recebendo com disponibilidade.

Ao professor Alexandre Falcão, por sua cordialidade e gentileza singulares e sua colaboração essencial na transposição de etapas difíceis. Levarei sempre comigo seu grande exemplo de Mestre, por sua disposição e paciência em sempre ensinar.

Aos amigos Marcos Quinet e Jacques, companheiros desde a graduação e que não mediram esforços em passar seus conhecimentos em muitas dúvidas ocorridas. Aqueles amigos que fiz no IC neste período e que estarão pra sempre comigo, Erick, Nelson, John, Johnny, Marcelo, Renata e tantos outros, por quem torço muito pelo seus sucessos.

Resumo

A Transformada Imagem-Floresta (IFT) foi proposta para a construção de operadores baseados em conectividade, reduzindo problemas de processamento de imagens baseados em conectividade em um problema de floresta de caminhos de custo mínimo em um grafo derivado da imagem. Sua utilização tem sido feita com sucesso para filtragem, análise e segmentação de imagens. Este trabalho tem seu foco na segmentação de imagens utilizando IFT, sobretudo na segmentação para detecção de bordas. Neste caso, a partir de uma seqüência ordenada de pontos selecionados sobre a borda dos objetos, a IFT detecta o contorno ótimo que passa por esses pontos. Porém, verificamos os três seguintes problemas: o método é sensível a localização dos pontos selecionados sobre a borda do objeto, bem como requer que estes estejam ordenados no sentido horário (ou anti-horário). Quando existem objetos muito próximos entre si e com características de imagem similares, alguns segmentos ótimos (arestas espúrias) podem grudar na borda dos outros objetos, ou seja, ocorre o aparecimento de arestas espúrias. Neste trabalho, nós analisamos estes problemas e possíveis soluções que dependem da aplicação. Particularmente, nós selecionamos uma aplicação de detecção de múltiplos objetos com texturas similares onde os três problemas ocorrem. Os resultados com as soluções analisadas/propostas melhoram substancialmente o desempenho da IFT.

Abstract

The Image Foresting Transform has been proposed for the design of image operators based on connectivity. The IFT reduces image processing problems into a minimum-cost path forest problem in a graph derived from image. It has been successfully used for image filtering, analysis and segmentation. In this work, we concentrate our research in the IFT image segmentation, especially the boundary tracking segmentation. In this case, from a sequence of points in order selected on the edge of objects, the IFT detects the optimun contour that passes for these points. However, we verify the three following problems: the method is sensible the localization of the points selected on the edge of the object, as well as it requires that these are ordered in the clockwise one (or counter-clockwise). When exists very next objects between itself and with similar characteristics of image, some optimun segments can stick in the edge of other objects, or either, it occurs the appearance of spurious edges. In this work, we analyze these problems and possible solutions that depend on the application. Particularly, we select an application of multiple object detention with similar textures where the three problems occur. The results with the analysed/proposed solutions substantially improve the performance of the IFT.

Palavras-chave

1. Processamento de Imagens
2. Segmentação de Imagens
3. Transformada Imagem-Floresta
4. Perseguição de bordas de objetos
5. Problemas de conectividade de pixels em imagens

Sumário

Lista de Figuras

Capítulo 1

Introdução

O processamento digital de imagens tornou-se imprescindível para alguns setores, tais como, análise de imagens biomédicas, meteorologia por meio de imagens de satélites, reconhecimento de padrões, restauração de pinturas antigas, análise de recursos naturais, monitoramento de poluição, cartografia, geologia, análise de imagens espaciais e controle de queimadas.

Uma das etapas mais importantes do processamento digital de imagens é a segmentação de imagens. A técnica de segmentação consiste em dividir a imagem em diferentes regiões, ou reparando objetos através de sua fronteira. Existem diversas técnicas de segmentação de imagens, mas não existe nenhum método que segmente de modo satisfatório todo tipo de imagem [3]. Há um conjunto de técnicas e a escolha de uma técnica em particular depende de fatores diversos como:

- Natureza da imagem : iluminação heterogênea, reflexos, presença de ruído, zonas texturizadas, contornos tênues ou parcialmente oclusos
- Operações realizadas após a segmentação : localização, medidas e cálculos 3D, reconhecimento de formas e interpretação, diagnóstico e controle de qualidade
- Primitivas a extrair : contornos, segmentos de reta, ângulos, regiões, formas e texturas
- Restrições do sistema : complexidade algorítmica, funcionamento em tempo real, tamanho da memória disponível, etc.

Dada esta diversidade, é difícil definir, de maneira absoluta, o que constitui uma boa segmentação. A segmentação não é um fim em si, sua qualidade é função dos tratamentos realizados, *a posteriori*, que utilizam as primitivas extraídas [3].

Após a segmentação é importante a identificação de objetos na imagem. Definimos objeto uma entidade individual que pode ser expressa por suas características para um determinado interesse específico. Existem diversas técnicas para este fim e muitas apresentam resultados satisfatórios e são bastante utilizadas, por exemplo transformada Watershed [5], contornos deformáveis [6, 7, 8, 9], level sets [10], porém pesquisas relacionadas ao tema e publicações ainda estão sendo realizadas pela necessidade cada vez mais crescente de técnicas eficientes de segmentação de imagens e para atender ao número elevado de aplicações de imagens digitais, por exemplo, em medicina, arqueologia, artes, área militar, segurança, dentre outras.

Neste contexto, a Transformada Imagem-Floresta (IFT) [2] é uma técnica desenvolvida recentemente por A. Falcão, J. Stolfi e R. A. Lotufo, pesquisadores da UNICAMP, que utiliza a representação de pixels de uma imagem e sua relação de adjacências para reduzir problemas de processamento de imagens baseados em conectividade em um problema de floresta de caminhos de custo mínimo (caminhos ótimos). Embora comparando com outras técnicas ainda são poucas as publicações sobre IFT, estas já são suficientes para mostrar o potencial da nova técnica para diversas aplicações em processamento de imagens [11, 12, 13, 14].

Através do Algoritmo de Dijkstra estendido, a IFT associa a cada pixel da imagem um caminho de custo mínimo, particionando a imagem em uma floresta de caminhos ótimo onde cada árvore tem como raiz um pixel semente e, como nós os pixels da imagem mais "conexos" com a raiz do que qualquer outra semente, em algum sentido apropriado.

Dependendo do problema a ser resolvido pela IFT, a função que calcula os custos mínimos será adaptada para atender aos objetivos da solução, ou seja, para cada problema, uma função de custo, as quais se baseiam em propriedades locais da imagem, como cor, gradiente, brilho, dentre outros.

Neste trabalho, buscamos tornar a compreensão da IFT mais acessível e para isso, estudamos os fundamentos da IFT e nos detemos com um pouco mais de detalhes na utilização da IFT para segmentação de imagens.

1.1 Problema

Durante o estudo e experimentos utilizando a IFT para a identificação de objetos, onde utilizamos determinadas imagens de fragmentos arqueológicos com características específicas, observamos o aparecimento de arestas espúrias quando dois objetos que se deseja

traçar seus contornos estão muito próximos, como mostrado na figura 1.1. Denominamos arestas espúrias quando verificamos que, no ponto onde estes objetos se aproximam ao seu máximo, os contornos traçados de ambos parecem desprenderem da borda dos mesmos, atestando um dos problemas da aplicação da segmentação para detecção de contornos pela IFT. Na figura 1.1, ilustramos um exemplo do aparecimento dessas arestas.

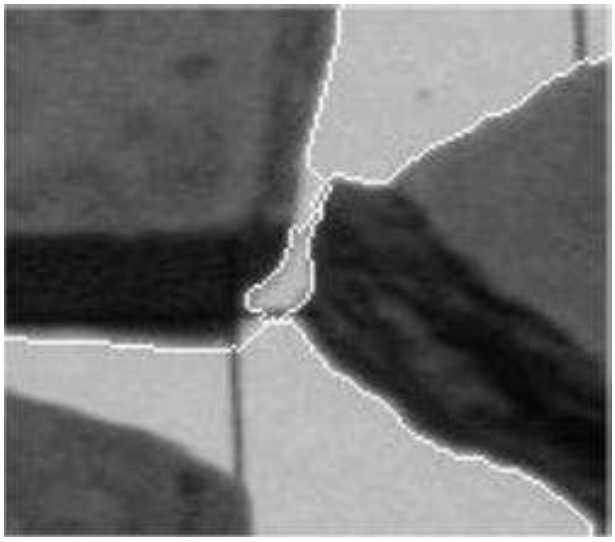


Figura 1.1: Aparecimento de arestas espúrias

No ponto onde os dois objetos estão mais próximos, podemos observar que os contornos traçados para ambos não acompanha a borda do objeto, permitindo inclusive que se coincidam, o que obviamente é incorreto.

Dois outros problemas são identificados, que são: o problema da ordenação dos pixels sementes e a importância da utilização destes mesmos pixels em pontos de alta curvatura da borda do objeto. A resolução do problema da ordenação por si só resolve o problema das arestas espúrias em muitas outras aplicações, ou seja, em outras amostras de imagens, porém, em nosso caso, necessita-se ainda de se realizar um pré-processamento inicial.

1.2 Objetivos do trabalho

Um dos objetivos deste trabalho é apresentar a técnica da IFT, com ênfase na segmentação de imagens, ampliando sua compreensão e tornando-a mais acessível a outros estudantes e pesquisadores. Da mesma forma, procuramos avaliar a técnica original para a detecção de contornos pela IFT, conforme apresentado na seção anterior onde, dependendo da imagem, podemos ter arestas espúrias. Analisando todo o processo de segmentação por

IFT, conseguimos detectar as causas da ocorrência deste problema citado da seção anterior e propomos uma solução.

Além desses objetivos, apresentamos algumas técnicas recentes de detecção de contornos de objetos mostrando suas vantagens e desvantagens. Procuramos assim, justificar a adoção da IFT neste trabalho e apresentar também o seu potencial para diversas aplicações, bem como suas vantagens em detrimento àquelas técnicas.

1.3 Organização do trabalho

O presente trabalho está dividido da seguinte maneira: no capítulo 2 que se segue, abordamos a segmentação de imagens e suas principais técnicas, com um enfoque maior àquelas de maior interesse nesta pesquisa. Apresentamos no capítulo 3 as principais propriedades topológicas de uma imagem, bem como sua representação através de um grafo. Nesta capítulo também apresentamos os principais conceitos em relação a Teoria dos Grafos, importantes para uma melhor compreensão da teoria e conceitos da IFT. No capítulo 4, apresentamos a técnica da IFT, teoria envolvida, seus algoritmos e exemplos de sua aplicação, com uma ênfase maior a perseguição de bordas. À partir da perseguição de bordas por IFT, apresentamos no capítulo 5 a aplicação realizada neste trabalho, bem como os problemas identificados e as soluções propostas. Os resultados obtidos e análises destes são apresentados no capítulo 6. Finalmente, no capítulo 7, expomos as considerações finais da presente dissertação.

Capítulo 2

Segmentação de Imagens

Neste capítulo, apresentaremos o problema de segmentação de regiões ou extração de objetos em imagens digitais. Embora muito estudado na literatura [15, 16, 17, 18, 19, 20, 21], não há um método único para separar regiões e/ou objetos em qualquer imagem, pois este depende das características da imagem e do que se deseja obter de resultado. A própria definição de uma região ou mesmo do objeto possui ambigüidades que precisam ser resolvidas para a implementação de um método. Apresentaremos neste capítulo os principais métodos baseados em região [15, 16, 21, 22, 23, 24] e em fronteira [16, 25, 18, 20, 26, 27].

2.1 Definição de imagem digital

Nesta seção, conceituamos imagem digital apresentando algumas características importantes para compreensão do restante deste trabalho.

Uma imagem é a representação de uma cena adquirida com a ajuda de sistemas de produção de imagens, tais como máquinas fotográficas digitais, vídeos, sensoriamento remoto, dentre outros. Desta forma, podemos definir, então, uma imagem digital $\hat{I}$ como sendo um par $(\mathcal{I}, \vec{I})$, onde $\mathcal{I}$ é um conjunto de pontos Z^n (domínio da imagem), denominados spels (*space elements*), e $\vec{I}$ é um mapeamento vetorial que associa a cada spel p em $\mathcal{I}$ um conjunto $\{I_1(p), I_2(p), ..., I_k(p)\}$ de valores escalares, associados com alguma propriedade física. O valor de n refere-se à dimensão da imagem e o valor de k ao número de propriedades associadas a p.

Uma imagem $\hat{I} = (\mathcal{I}, I)$ em tons de cinza e bidimensional ($\mathcal{I} \subset Z^2$) possui apenas uma banda $I(k = 1)$, onde os spels são chamados de pixels (*picture elements*).

A imagem é portanto uma matriz de tamanho $N \times M$ pixels (N linhas e M colunas).

Sua representação vetorial relaciona o índice i a cada pixel $p = (x, y)$ por:

$$i = x + M * y$$

$$x = i\%M$$

$$y = i/M$$

Os valores $I(p)$ de cada pixel p são obtidos por amostragem e quantização de uma função contínua $I_c(x, y)$ que descreve a propriedade física correspondente em uma dada região do espaço. No caso de uma foto temos o brilho, e no caso de uma tomografia de Raios-X, temos a densidade do tecido.

2.1.1 Vizinhança

Em uma imagem digital $\hat{I} = (\mathcal{I}, I)$ no domínio de Z^2, um pixel $p = (x, y)$ tem quatro vizinhos que compartilham uma aresta com $p : p = (x+1, y), (x-1, y), (x, y+1)$ e $(x, y-1)$ (figura 2.1). Este conjunto é chamado de vizinhança 4 de p ($N_4(p)$).

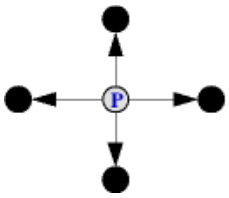


Figura 2.1: Vizinhança 4 de um pixel

Considerando os vizinhos que compartilham pelo menos um vértice com $p = (x, y)$ temos um conjunto vizinhança 8 de p ($N_8(p)$) (figura 2.2). Este conjunto é formado pelos pixels de $N_4(p)$ e os pixels diagonais $(x+1, y+1), (x-1, y+1), (x+1, y-1)$ e $(x-1, y-1)$. Um tratamento especial é normalmente dado aos pixels que pertencem às bordas da imagem, pois alguns de seus vizinhos vão estar fora da imagem.

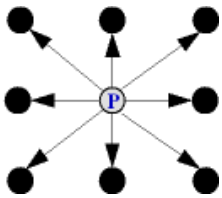


Figura 2.2: Vizinhança 8 de um pixel

Outras propriedades topológicas importantes sobre imagens serão devidamente abordadas no capítulo 3.

2.1.2 Conexidade entre pixels

Para que dois pixels sejam ditos conexos, eles devem ser "vizinhos" e ambos devem satisfazer um critério comum de similaridade, tais como brilho, cor, posição, dentre outros. A conexidade é normalmente utilizada quando desejamos traçar contornos, isto é, definir bordas do objeto em uma imagem. Maiores detalhes sobre a conexidade entre pixels numa imagem serão abordados no capítulo seguinte.

2.2 Definição de segmentação de imagens

A segmentação de imagens [3] é definida como um tratamento que visa particionar o domínio de imagem $\hat{I} = (\mathcal{I}, I)$ em um subconjunto composto de n regiões R_i tais que:

$$0 < i \leq n \begin{cases} \forall i \ R_i \neq \emptyset \\ \forall i, j : i \neq j; R_i \bigcap R_j = \emptyset \\ \mathcal{I} = \bigcup_i R_i \end{cases} \tag{2.1}$$

De acordo com a definição 3.1 acima, qualquer região particionada na imagem $\hat{I}$ deve ser não-vazia, ou seja, deve possuir uma parte da região $\hat{I}$ (restrição 1). Quaisquer duas regiões i e j particionadas de $\hat{I}$, estas devem ser distintas, não permitindo áreas comuns entre as mesmas (restrição 2). Logicamente, se particionamos a imagem $\hat{I}$ em n regiões, a união destas formam a própria imagem $\hat{I}$ (restrição 3).

A segmentação precede as fases de extração de atributos da imagem, reconhecimento e interpretação [1], sendo, portanto, uma das primeiras fases da análise de imagens. Desta forma, ela desempenha um papel crítico na seqüência de tratamento da imagem, pois, caso erros sejam cometidos nesta fase, estes são propagados nas fases subseqüentes, influenciando para a obtenção de resultados ruins no processamento final das imagens [28].

Os métodos de segmentação são informalmente separados em duas classes: a dos *métodos baseados em bordas* e a dos *métodos baseados em regiões* [3]. Em última análise, o que se faz é considerar a *continuidade* ou a *descontinuidade* dos níveis de cinza da imagem. A detecção de bordas, por exemplo, explora propriedades relacionadas à sua descontinuidade, enquanto a limiarização concerne às propriedades relacionadas a sua similaridade [28]. Estes procuram extrair contornos fechados, onde um contorno fechado é um caminho de pixels 4- ou 8-adjacentes que separa o interior do exterior do objeto. Isto é, todo caminho 4-conexo vindo de dentro para fora, ou vice-versa, deve cruzar a borda

em um pixel. Se o caminho for 8-conexo, então deve cruzar a borda em pelo menos um vértice de pixel. Algumas abordagens também definem orientação para o contorno. Isto é, se caminharmos ao longo da borda teremos sempre um dos lados (direito ou esquerdo) no interior e o outro no exterior do objeto. Métodos baseados em região extraem o conjunto de pixels que representa o interior do objeto, incluindo os pixels de fronteira. Algumas abordagens classificadas como híbridas utilizam estratégias baseadas em bordas e em região simultaneamente, independente da representação final.

A segmentação de uma imagem $\hat{I} = (\mathcal{I}, I)$ baseada em região pode ser vista como um mapeamento que associa para todo pixel $p \in \mathcal{I}$ um inteiro $L(p)$, denominado rótulo, cujo valor é diferente para cada objeto (incluindo o fundo). Neste caso, a segmentação é dita *hard* porque cada pixel p só pertence a um único objeto. Algumas abordagens estendem este conceito para *fuzzy* [29], onde cada pixel p pertence a todos os objetos com diferentes graus de pertinência.

2.3 Técnicas baseadas em região

A forma mais simples de particionar uma imagem é dividí-la em duas regiões, objetos e fundo, aplicando uma classificação pixel a pixel. Esta classificação pode ser baseada em uma única característica (e.g. o brilho) ou em um conjunto de características (e.g. brilho, gradiente, matiz) dos pixels. A classificação também se aplica a múltiplas regiões (objetos) que formam algomerados (*clusters*) no espaço de caracteríticas. Exemplos dessas técnicas são limiarização (*thresholding*) [30], classificação estatística [20], redes neurais e *clustering* [31]. Entre essas, limiarização e clustering são as mais usadas para classificação de pixels e serão apresentadas a seguir.

2.3.1 Limiarização

Na limiarização, a imagem é entendida como uma região formada por pixels conexos que tenham atributos em comum[32]. O processo se baseia na análise do histograma da imagem, que é um gráfico que mostra o número de pontos de uma imagem que tem o mesmo nível de cinza. Na abscissa temos os níveis de cinza enquanto na ordenada temos a quantidade de pontos. Uma imagem digitalizada não colorida consiste de vários tons de cinza (ou no mínimo 2 tons, se for preto e branco).

Todos os pixels que estão dentro de uma faixa de intensidade são classificados como pertencentes a uma mesma região. Em sua forma mais geral a limiarização pode ser

descrita matematicamente como:

$$S(i,j) = k,\ se\ T_k - 1 \leq f(i,j) < T_k\ para\ k = 1, 2, ..., m$$

onde *S(i,j)* é a função resultante, *f(i,j)* é a função original (imagem), $T_0, \cdots, T_m$ são os valores de limiarização (thresholding) e m é o número de classes distintas a serem aplicadas à imagem. Se $m = 2$, o método de limiarização é denominado limiarização binária. Se, por outro lado, $m > 2$ o método é descrito como limiarização multi-modal.

A limiarização é sensível a irregularidades de fundo e variações locais da imagem [19, 32]. Portanto, na limiarização há o realce dos objetos da imagem, não ocorrendo necessariamente a identificação de objetos na mesma.

2.3.2 Clustering

Técnicas de clustering se baseiam no princípio de se agrupar um conjunto de dados (ou informações) em diferentes grupos com propriedades similares. Estas técnicas podem ser utilizadas para segmentação de imagens, onde tem-se que cada pixel $p \in \mathcal{I}$ de uma imagem $\hat{I} = (\mathcal{I}, I)$ tem associado um vetor de características de imagem, tais como brilho, variância e magnitude de gradiente. Para um número n de características, cada pixel p é mapeado em um ponto no espaço $\mathcal{R}^n$ de características [33]. Técnicas de clustering se baseiam na premissa que regiões de interesse para segmentação formam aglomerados de pontos separáveis em $\mathcal{R}^n$, e procuram identificar esses aglomerados particionando o espaço $\mathcal{R}^n$ em k regiões $R_1, R_2, ..., R_k$ tais que $\bigcap_{i=1}^{k} R_i = \emptyset$ e $\bigcup_{i=1}^{k} R_i = \mathcal{I}$. Se cada região R_i estiver associada a um único objeto $O_i, i = 1, 2, ..., k$ incluindo o fundo, então a partição obtida é a solução da segmentação. Caso contrário, quando um objeto possui múltiplas regiões de pixels com características distintas, temos ainda que associar quais regiões pertencem a quais objetos. Informações globais sobre o problema - tais como características das regiões, vizinhança entre regiões, e a forma gerada pela união de regiões - poderiam ser usadas para completar a segmentação.

O particionamento em regiões requer o cálculo de distância entre pontos $\mathcal{R}^n$. Esta distância pode ser traduzida em uma dissimilaridade $d(p, q)$ entre os pixels $p, q \in \mathcal{I}$ que esses pontos representam. Por exemplo, podemos ter em $\mathcal{R}^2$,

$$d(p,q) = [I(p) - I(q)]^2 + [G(p) - G(q)]^2,$$

onde G(p) é a magnitude de um gradiente (e.g. Sobel) calculado em p. Existem várias

funções de dissimilaridade, mas elas normalmente satisfazem os axiomas métricos:

1. $d(p,q) \geq 0,$ e se $d(p,q) = 0,$ então $p = q.$
2. $d(p,q) = d(q,p).$
3. $d(p,r) \leq d(p,q) + d(q,r).$

As regiões R_i são definidas dinamicamente durante o algoritmo de clustering. Isto permite que a dissimilaridade entre dois pixels varie durante o algoritmo e seja medida mais global do que local. Alguns variantes definem $d(p,q)$ como a dissimilaridade entre regiões R_i e R_j, onde $1 \leq i,j \leq k, i \neq j, p \in R_i$ e $q \in R_j$, levando-se em conta as características de todos os pixels que pertecem a esseas regiões no dado instante.

Clustering, portanto, agrupa pixels mas não identifica objetos, podendo ser utilizado com limiarização para segmentar imagens. Maiores detalhes sobre clustering podem ser encontrados em [33, 34]

Um aspecto importante na segmentação de imagens é a conexidade entre pixels que pertencem a um mesmo componente. As técnicas mencionadas acima não exploram a conexidade, exceto alguns algoritmos de clustering que podem ser aplicados na imagem usando a relação de adjacência em vez de serem aplicados no espaço de características. A conexidade é explorada em técnicas de crescimento de regiões e técnicas que usam divisão e conquista de regiões conexas. O crescimento de regiões usa um conjunto de pixels sementes marcadas em um critério de parada. Em algumas abordagens, as regiões crescem a partir de sementes marcadas em um único objeto até atingirem o critério de parada, idealmente na fronteira do objeto com o fundo. Outras abordagens usam sementes em vários objetos (incluindo o fundo) e o critério de parada passa a ser definido pelo choque entre as regiões.

2.4 Métodos Baseados em borda

Os métodos de segmentação baseados em bordas (ou arestas) localizam regiões da imagem onde a variação dos tons de cinza ocorre de maneira relativamente abrupta. As descontinuidades, como são chamadas, podem ocorrer na forma de pontos isolados, linhas, segmentos ou curvas e, a partir delas, são formadas as bordas dos objetos contidos na imagem [30]. Desta forma, podemos identificar os objetos em uma imagem, através de diversas métodos baseados em borda, conforme veremos a seguir.

De fato, a existência de tais descontinuidades é característica de um conjunto limitado de imagens. Em muitas delas, a transição de uma região para outra ocorre de maneira tão sutil que tornam a aplicação dos métodos de detecção de borda uma opção inviável. Após a detecção das descontinuidades segue-se, geralmente, a aplicação de algum método capaz de conectar tais fragmentos e gerar contornos que estejam associados com os contornos reais dos objetos.

A abordagem mais simples, portanto, é por classificação de pixels. Aplica-se um filtro de suavização para remoção de ruídos, seguido de um realce de bordas (e.g. magnitude de gradiente, Laplaciano) e depois uma classificação binária (borda/fundo) para decidir quais pixels pertencem à borda de um objeto. Critérios locais buscam por segmentos próximos a cada segmento que possam pertencer a mesma borda (e.g. limiarização por histeresis) e os critérios globais assumem que segmentos de uma mesma borda satisfazem uma dada equação (e.g. Transformada de Hough) [35].

Outra forma de abordar o problema é transformar a imagem em um grafo, e aplicar um algoritmo de busca por caminhos ótimos no grafo, onde cada caminho é um segmento de borda. Técnicas baseadas em busca heurística usando um determinado algoritmo específico e programação dinâmica são mais populares. Essas técnicas costumavam impor restrições topológicas e geométricas para a borda e não consideravam todos os arcos de modo que nem sempre garantiam uma solução. Como veremos mais adiante, o método da Transformada Imagem-Floresta(IFT) generaliza essas técnicas eliminando esses problemas.

Um aspecto negativo nas abordagens acima é a falta de informação global sobre o objeto no modelo de segmentação. Métodos baseados em **contornos deformáveis** procuram resolver o problema através de Equações Parciais e Diferenciais (PDE). A idéia é partir de um contorno inicial que deforma-se para minimizar um funcional de energia, o qual deve ser mínimo quando o contorno adere à borda do objeto. Na maioria das técnicas, porém, a informação relevante para extrair o objeto não é incorporada no funcional de energia e o método falha na segmentação.

2.4.1 Transformada de Hough

O princípio básico da transformada de Hough é mapear os pixels classificados como borda para um espaço de parâmetros de uma dada equação. Pixels que pertencem a segmentos de broda e que satisfazem a esta esquação para um dado conjunto de parâmetros devem formar um aglomerado de pontos no espaço de parâmetros. A localização deste aglo-

merado de pontos permite identificar na imagem de bordas quais pixels pertencem à borda de interesse, eliminando os segmentos espúrios. A própria equação pode ser usada para fechar os buracos entre os segmentos selecionados.

O caso mais simples é a identificação de segmentos de reta. Todos os pixels que satisfazem a equação de uma reta $y = a'x + b'$ são mapeados no ponto (a', b') do espaço de parâmetros aXb. Como não conhecemos a' e b', a idéia é quantizar o espaço de parâmetros para possíveis valores de a e b, depois acumular em cada posição (a, b) o número de pixels e quais pixels satisfazem a equação $y = ax + b$. Os pixels da reta, portanto, formarão um aglomerado mais alto neste histograma bidimensional em torno de (a', b'). Um problema, porém, é que a e b assumem valores infinitos para retas verticais. Demais detalhes podem ser vistos em [27, 35].

2.4.2 Contornos Deformáveis

Contornos deformáveis têm se tornado bastante populares nos últimos 10 anos [6, 7, 8, 9, 36, 37, 38, 39]. Entre estes modelos, o mais popular é conhecido por método *snakes*, desenvolvido por M. Kaas et al. no final da década de 80 [40]. *Snakes* são curvas planares abertas ou fechadas (e.g. splines de continuidade controlada) que têm associado um funcional de energia e que se deformam sob a ação de forças internas e externas com o objetivo de minimizar este funcional. A idéia básica é que a partir de uma localização inicial na imagem, a *snake* se deforme até atingir um valor mínimo de energia que deve coincidir com a situação em que a *snake* adere à borda do objeto desejado na imagem. Um contorno ativo é um caso particular de *snakes*, onde a curva é fechada. Este modelo é usado para representar uma borda 2D de um objeto na imagem. Também existem extensões deste modelo para segmentar objetos 3D [6, 41, 42] a partir de um conjunto de fatias tomográficas. No trabalho de Rodrigues et. al [43] pode-se encontrar mais detalhes sobre contornos ativos no modelo discreto e contínuo.

Uma das principais dificuldades dos modelos de contornos deformáveis é encontrar a melhor localização para o contorno inicial. Se este não estiver próximo ao objeto de interesse, ele não é atraído para a borda do objeto podendo se reduzir a um ponto na imagem [43]. Outras dificuldades incluem: bordas fracas podem levar a comportamentos instáveis; o processo de minimizar a energia pode levar a oscilações desnecessárias; a escolha dos parâmetros do modelo tem um forte efeito sobre seu comportamento; o contorno fica preso em mínimos locais; o contorno tem problemas para aderir à borda do objeto, etc.

Estas dificuldades têm sido a principal motivação de muitos trabalhos sobre contornos deformáveis [44, 45]. A estabilidade da snake tem sido tratada pelo ajuste de parâmetros internos [46]. Leymarie e Levine [47] introduziram limites na força da imagem, novas regras para estabelecer parâmetros de rigidez e uma nova condição de término para a curva. Eles usaram snakes para representar a forma de seus objetos planos usando a transformada de distância para minimizar a função de energia [36], e para tratar objetos não rígidos, e.g. células [47]. Amini et al. [48] usou programação dinâmica para melhorar a estabilidade. Neunschwande et al. [39] apresentou um método no qual o usuário somente especifica os pontos finais da curva inicial. Zucker integrou evidência de borda local obtida através de um procedimento de relaxamento de rótulos, usando snakes que poderiam gerar estruturas construídas a partir destas bordas rotuladas [49]. Berger and Mohr [46] também usaram várias pequenas snakes crescentes para detectar comportamentos locais. Cooper et al. [50] apresentou um modelo probabilístico chamado "ripple filter", o qual age como um contorno deformável. Apesar destas melhorias, um número de dificuldades consideráveis ainda permanecem, em particular, snakes sempre recaem na questão de uma propriedade de inicialização próxima da borda, além de requerer múltiplas inicializações, uma por objeto de interesse, etc.

2.5 Técnicas híbridas

Existem várias formas de combinar técnicas baseadas em região com técnicas baseadas em borda. Uma idéia interessante é modelar a fronteira das regiões em abordagens de crescimento de regiões como um contorno deformável, fazendo com que o funcional de energia influencie no crescimento das regiões. O crescimento de regiões também poderia ser aplicado para inicializar o contorno deformável próximo à borda desejada, ou podemos ainda aplicar dois contornos deformáveis de dentro para fora e de fora para dentro do objeto como sementes inicias para o crescimento de regiões [8, 43].

2.6 Segmentação Usando o Algoritmo *Watershed*

O conceito da transformada divisor de águas (*Watershed*) foi originalmente proposto por Digabel e Lantuéjoul [51] e posteriormente elaborado por Beucher e Lantuéjoul [52]. A idéia básica consiste em observar uma imagem digital em tons de cinza como se ela fosse uma superfície montanhosa, sendo que a altitude de cada ponto está diretamente relacionada ao nível de cinza do pixel correspondente. Uma gota de água que cair sobre

esta superfície irá percorrer o caminho mais íngreme até chegar a uma região de mínimo. O conjunto de todos os pontos (pixels) para os quais uma gota de água que cai e converge para a mesma região de mínimo é chamada represa (catchment basin). Porém, para alguns pontos não é possível determinar para onde irá escorrer a gota de água que ali cair. O conjunto destes pixels formam as fronteiras das represas e são denominados divisores de água (*Watershed*), como ilustrado na Figura 2.3

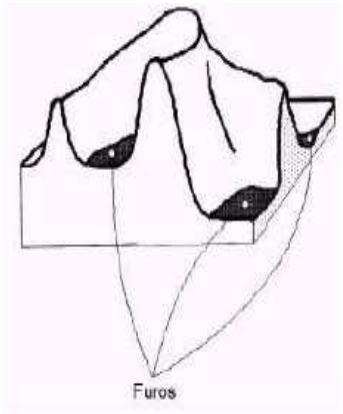


Figura 2.3: Ilustração da topografia com os pontos de mínimo [1].

A transformada divisor de água é normalmente aplicada sobre a transformada gradiente de uma imagem. Isto porque os contornos da imagem serão reforçados após a aplicação do gradiente, procuranado, assim, melhorar o processo de segmentação.

Capítulo 3

Imagem como um grafo

Neste capítulo identificamos algumas propriedades importantes de imagem e mostramos como uma imagem pode ser interpretada como um grafo.

3.1 Propriedades topológicas de uma imagem

Topologia digital é o estudo de propriedades de objeto em imagem digital, as quais não são afetadas por transformações geométricas, exceto aquelas que envolvem junção ou separação de partes do objeto. Em processamento de imagem digital, um objeto em uma imagem 2D (ou 3D) é aproximado por um conjunto de pixels (ou voxels). Portanto, topologia digital estuda as propriedades deste conjunto de pixels (ou voxels) que correspondem às propriedades topológicas do objeto original. Nesta seção nossa meta é definir apenas os conceitos básicos usados em topologia digital para compreender a seqüência deste trabalho.

3.1.1 Relação Binária

Uma relação binária $\mathbf{R}$ aplicada a um conjunto $\mathbf{X}$ é um subconjunto do produto cartesiano $\mathbf{X} \times \mathbf{X}$. Uma relação binária é dita reflexiva se $(a, a) \in \mathbf{R}$, para todo $a \in \mathbf{X}$, simétrica se $(a, b), (b, a) \in \mathbf{R}$, para todo $a, b \in \mathbf{X}$, e transitiva se $(a, b), (b, c) \in \mathbf{R}$ implica que $(a, c) \in \mathbf{R}$, para todo $a, b, c \in \mathbf{X}$. Neste caso $\mathbf{R}$ é dita de equivalência [31].

3.1.2 Relação de Adjacência e Grafos

Uma relação de adjacência $\mathcal{A}$ é uma relação binária entre pixels, a qual depende de suas posições, e opcionalmente de outras propriedades locais da imagem [31]. Dizemos que

$\mathcal{A}(p)$ é o conjunto dos pixels adjacentes ao pixel p de acordo com $\mathcal{A}$. Isto é, $q \in \mathcal{A}(p)$ é o mesmo que $(p,q) \in \mathcal{A}$. Uma relação de adjacência leva, portanto, à definição de um grafo $\mathbf{G} = (D, \mathcal{A})$ para a imagem, onde D é o conjunto de pixels. Neste grafo, um caminho π é uma seqüência de pixels adjacentes $< p_1, p_2, ..., p_n >$, onde $(p_i, p_{i+1}) \in \mathcal{A}, i = 1, 2, ..., n-1$. Exemplos:

1. $(p,q) \in \mathcal{A}$ se $d(p,q) <= \rho$, onde d é distância Euclideana e ρ é um escalar,
2. $(p,q) \in \mathcal{A}$ se $q - p \in \{(-1,-1),(1,-1)\}$,
3. $(p,q) \in \mathcal{A}$ se $(x_p - x_q) + (y_p - y_q) \leq 1$ e $(f(p) - f(q)) \leq l$, onde l é um limiar de brilho.

Observe que $\rho = 1$ é vizinhança-4, $\rho = \sqrt{2}$ é vizinhança-8, e $\rho = \sqrt{5}$ faz com que pixels mais distantes sejam vizinhos no grafo. Esta relação é simétrica e invariante à translação. Note também que o segundo exemplo está relacionado com a definição de elemento estruturante planar usada em morfologia matemática, e portanto uma relação de adjacência pode ser assimétrica.

3.1.3 Relação de Conexidade

Um pixel p é conexo a um pixel q se existir um caminho de p a q no grafo definido por $\mathbf{A}$. a conexidade pode ser assimétrica [53].

3.1.3.1 Componente Conexo

Um componente conexo de um conjunto $\mathbf{X}$ de pixels é um subconjunto $\mathbf{Y} \subset \mathbf{X}$, onde todos os pares (p,q) de pixels em $\mathbf{Y}$ são conexos (i.e. existe um caminho de p a q e um caminho de q a p , que não necessariamente são os mesmos)[31]. Por exemplo, seja $\mathbf{R}$ uma relação binária "conexidade-4" aplicada sobre o conjunto $\mathbf{X} = \{p_1, p_2, p_3, p_4\}$ da figura 3.1; isto é $\mathbf{R} = \{(p_1,p_2),(p_2,p_1),(p_2,p_3),(p_3,p_2),(p_1,p_3),(p_3,p_1)\}$. Podemos notar que não é conexo-4 com nenhum outro elemento de $\mathbf{X}$.

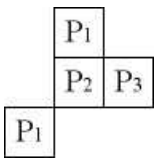


Figura 3.1: O conjunto $\mathbf{X}$ possui dois componentes conexos-4: $\{p_1, p_2, p_3\}$ e $\{p_4\}$

3.1.3.2 Identificação de objetos em uma imagem

Mencionamos antes que a segmentação de imagens é utilizada para identificar um objeto do resto dos pixels da imagem. Considerando o exemplo da Figura 3.2, podemos particionar uma imagem em componentes conexos disjuntos de acordo com diferentes critérios de classificação.

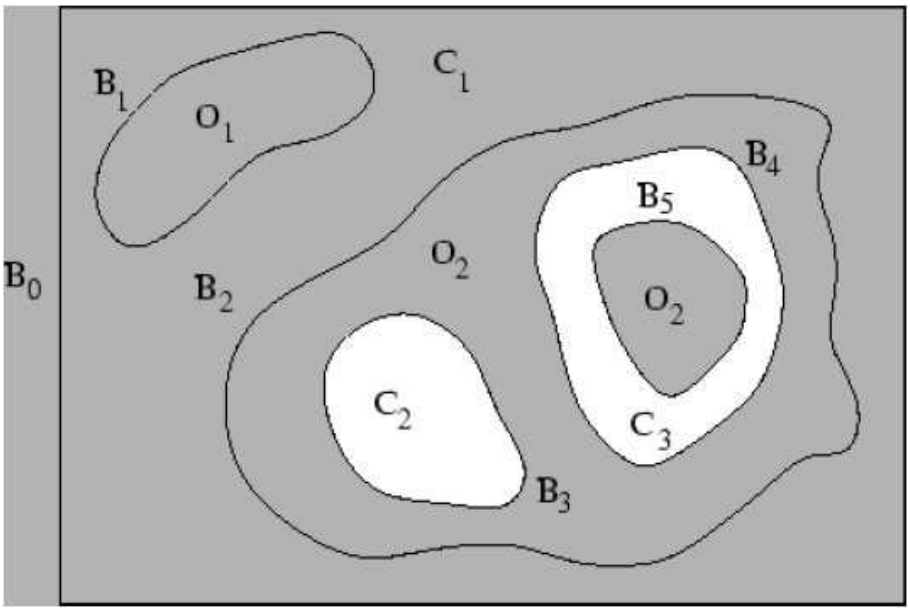


Figura 3.2: Uma imagem e seus componentes conexos

Desta forma, o fundo da imagem é definido por um ou mais componentes conexos, C_1, C_2 e C_3, que não têm significado de interesse para a aplicação. Cada objeto pode ser definido como um ou mais componentes conexos cuja vizinhança é composta por componentes do background ou por outros objetos. Neste caso temos dois objetos, O_1 e O_2, onde o primeiro possui um componente conexo e o segundo dois. Uma borda de objeto é um conjunto de pixels do seu interior que possui ao menos um pixel adjacente no exterior. As bordas do objeto O_2, por exemplo, são B_2, B_3, B_4 e B_5. Um objeto pode ser representado por suas bordas ou pelos pixels que compõem seu interior.

3.2 Grafos e conceitos relacionados

Como um dos objetivos deste trabalho é o estudo da segmentação de imagens por IFT para identificar objetos, devemos previamente abordar alguns conceitos relacionados a grafos para melhor compreensão do funcionamento da IFT.

3.2.1 Definição de grafos

Quando analisamos um conjunto de elementos de dados (não necessariamente dados computacionais), podemos estar preocupados com o seu conteúdo ou com as relações existentes entre eles. O grafo propriamente dito é uma representação gráfica das relações existentes entre elementos de dados. Ele pode ser descrito num espaço euclideano de n dimensões como sendo um conjunto **V** de *vértices* e um conjunto **A** de curvas contínuas (*arestas*).

Formalmente, um grafo **G** é definido por um par ordenado de conjuntos **(V, E)**, e escrevemos **G = (V, E)**, tal que **V** é finito e cada elemento de **E** é um subconjunto formado por dois elementos de **E**. Um elemento de **V** é chamado de vértice do grafo **G** e um elemento de **E** é chamado de aresta de **G**. Para evitar ambigüidades supomos que $\mathbf{V} \cap \mathbf{E} = \emptyset$. Por exemplo, os conjuntos $\mathbf{V} = \{A, B, C, D, E, F, G\}$ e $\mathbf{E} = \{ (A, B), (A, E), (B, E), (C, D), (E, G)\}$ definem um grafo [54], como mostrado na figura 3.3 abaixo:.

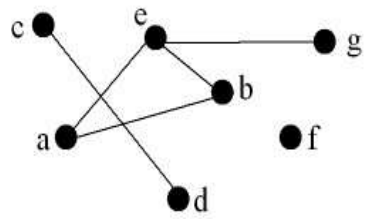


Figura 3.3: Representação geométrica (ou diagrama) do grafo

Todo grafo pode ser representado geometricamente desenhando um ponto no plano para cada vértice e um segmento de curva ligando cada par de vértices que formam uma aresta. Claramente, a representação geométrica de um grafo não é única. Quando uma curva possui indicação de sentido (uma seta), ela é chamada de *arco*, caso contrário é chamada de *linha*.

Num grafo **G**, um vértice v incide na aresta t se v e t. Dois vértices u e v são adjacentes se $\{u, v\} \in E(G)$. Duas arestas t e f são adjacentes se $|t \cap f| = 1$, isto é, têm um vértice em comum. Quando $t = \{v, u\}$ dizemos que u e v são os extremos da aresta t. Como outras definições importantes sobre grafos [55], temos:

Grafos orientados: Num grafo orientado as arestas têm sentido de modo que se u e v são vértices, então $uv \neq vu$ e além disso se uv é uma aresta então vu não é aresta. Removendo o sentido das arestas temos o grafo subjacente ao grafo orientado.

Grafos direcionados ou dirigidos: Num grafo direcionado as arestas têm sentido

e formam um multiconjunto. Ou seja, um grafo direcionado é uma par $(\mathcal{I}, \mathcal{A})$ onde $\mathcal{I}$ é um conjunto de nós e $\mathcal{A}$ é um conjunto de pares ordenados de nós. Um exemplo de grafo direcionado é apresentado na figura 3.4 a seguir:

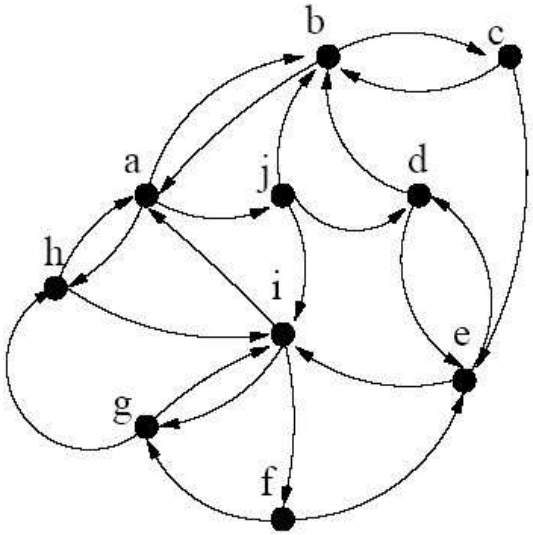


Figura 3.4: Grafo direcionado

Grafo valorado Um grafo valorado ou com peso nas arestas é um tripla $(\mathbf{V}, \mathbf{E}, \rho)$ de modo que **(V, E)** é um grafo e $\rho : E \rightarrow R$ é uma função que assume valores em $R \subset \mathbb{R}$. O conjunto R em geral depende do problema sendo considerado. Por exemplo, se os pesos representam distância então tomamos $R = \mathbb{R}^+$(reais não-negativos).

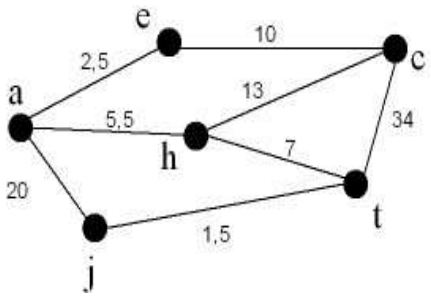


Figura 3.5: Grafo valorado

3.2.2 Caminhos

Um caminho é uma seqüência de vértices tal que de cada um dos vértices existe uma aresta para o vértice seguinte [54]. Um caminho é chamado **simples** se nenhum dos vértices no

caminho se repete.

Formalmente, portanto, um caminho é uma seqüência $< t_1, t_2, \ldots, t_k >$ de nós distintos em um grafo, tal que (t_i, t_{i+1}) pertence a $\mathcal{A}$ para $1 \leq i \leq k-1$ [53]. Denotamos a origem t_1 e o destino t_k de π, por $org(\pi)$ e $dst(\pi)$, respectivamente. O caminho é trivial se $k = 1$. Se π e τ são caminhos tais que $dst(\pi) = org(\tau) = t$, denotamos por $\pi \cdot \tau$ a concatenação dos dois caminhos (Figura 3.6).

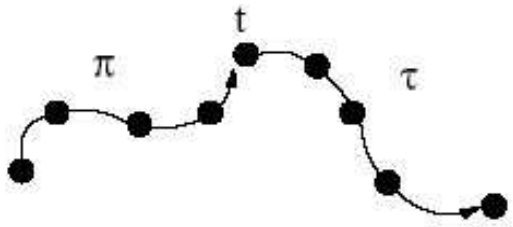


Figura 3.6: Caminho em grafo

Denotamos $\pi \cdot \tau < s, t >$ a concatenação do antecessor mais distante τ de π e o último arco (s, t).

3.2.2.1 Caminhos Mínimos

O problema do caminho mínimo é bastante conhecido e tem como objetivo obter um percurso mínimo entre dois ou mais vértices de um grafo [53]. Neste caso, um grafo pode representar uma malha rodoviária, distâncias geográficas ou, no caso deste trabalho, as conexões entre pixels na imagem. A determinação dos menores caminhos aparece constante e consistentemente como um subproblema de problemas mais complexos em grafos.

O problema do caminho mínimo se adapta a diversas situações práticas. Em roteamento, por exemplo, pode-se modelar os nós do grafo como cruzamentos, as arestas como vias, e os custos associados às arestas correspondem a tempo de trajeto ou distância percorrida, e a solução será o caminho mais curto ou o caminho mais rápido entre dois pontos. Outras possibilidades de aplicação incluem quaisquer problemas envolvendo redes ou grafos em que se tenham grandezas (distâncias, tempo, perdas, ganhos, despesas) que se acumulem linearmente ao longo do percurso do grafo.

3.2.2.2 Árvore de custo mínimo

O Problema da Árvore de Custo Mínimo é encontrar uma árvore que contenha todos os vértices do grafo e cuja soma das arestas seja mínima [53, 55, 56].

Este problema aparece, por exemplo, no seguinte contexto: é dado um mapa de n cidades rurais com uma matriz listando as distâncias euclidianas entre todos os pares possíveis de cidades e deseja-se obter o menor comprimento de rodovias necessárias para unir todas elas. Outro contexto importante seria para auxiliar na decisão de onde se localizar postos de emergência ou delegacias de polícia em uma cidade, por exemplo. Além dessas aplicações, a solução desse problema é de grande utilidade para a solução de outros problemas mais complexos em grafos, tais como o do caixeiro viajante e, no nosso trabalho, aplicá-lo a um mapa de pixels que define uma imagem.

Dado um grafo onde as arestas tem valores numéricos (os custos), podemos definir uma forma que avalie o custo de um caminho num grafo.

3.2.3 Funções de custo de caminhos

Uma função de custo de caminho é uma mapa que assume para cada caminho π, um custo $f(\pi)$, em algum conjunto ordenado $\mathcal{V}$ de valores de custo [53].

Uma função f é dita **monotônica-incremental (MI)** quando:

$$\begin{aligned} f(< t >) &= h(t), \\ f(\tau \cdot < s, t >) &= f(\tau) \odot (s, t) \end{aligned} \tag{3.1}$$

onde $h(t)$ é um handicap de valor de custo e $\odot$ satisfaz:

$\boldsymbol{x' \geq x \Rightarrow x' \odot (s,t) \geq x \odot (s,t)}$ e $\boldsymbol{x \odot (s,t) \geq x}$, para $x, x' \in \mathcal{V}$ e $(s,t) \in \mathcal{A}$.

Como exemplos de funções de custo MI [56], temos:

1. Função de custo Aditiva : computa o somatório dos pesos dos arcos no caminho.

$$\begin{aligned} f_{sum}(< t >) &= h(t), \\ f_{sum}(\pi \cdot < s, t >) &= f_{sum}(\pi) + w(s, t) \end{aligned} \tag{3.2}$$

   onde $w(s,t)$ é um peso fixado para o arco, não-negativo.

2. Função de custo Max-arco : computa o valor máximo entre todos os pesos dos arcos no caminho.

$$f_{max}(< t >) = h(t), \tag{3.3}$$
$$f_{sum}(\pi \cdot < s, t >) = max\{f_{max}(\pi), w(s,t)\}$$

onde $w(s,t)$ é um peso fixado para o arco.

Para exemplificarmos o emprego destas funções, seja a figura 3.7 a seguir contendo um grafo direcionado:

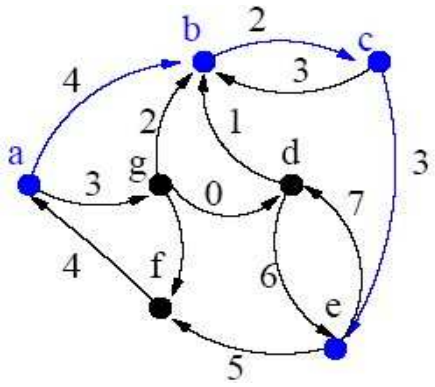


Figura 3.7: Exemplo do emprego das funções MI

Considerando $h(a) = 1$, teremos $f_{max}(< a, b, c, e >) = 4$ e $f_{sum}(< a, b, c, e >) = 10$. Numa segunda opção, considerando $h(a) = 5$, teremos $f_{max}(< a, b, c, e >) = 5$ e $f_{sum}(< a, b, c, e >) = 14$.

Um caminho π é *ótimo* se $f(\pi) \leq f(\pi')$ para qualquer outro caminho π' com $dst(\pi') = dst(\pi)$, independente de seus pontos iniciais. Neste caso, $f(\pi)$ é por definição o custo do pixel $t = dst(\pi)$, denotado por $\hat{f}(t)$. Podemos notar que um caminho trivial não é necessariamente ótimo: mesmo para f_{sum} ou f_{max}, os handicaps de dois pixels s e t podem ser tal que um caminho não-trivial de s para t seja mais barato que o caminho trivial $< t >$

3.2.4 Mapa de predecessores e floresta espalhada

Um mapa de predecessores é uma função P que atribui a cada nó $t \in \mathcal{I}$ algum outro nó em $\mathcal{I}$, ou um marcador distinto $nil \notin \mathcal{I}$ - neste caso, t é a raiz do mapa [2].

Uma floresta espalhada é um mapa de predecessores que leva todo nó para *nil* em um número infinito de iterações (i.e. não contém círculos), como exemplificado na figura 3.8 a seguir:

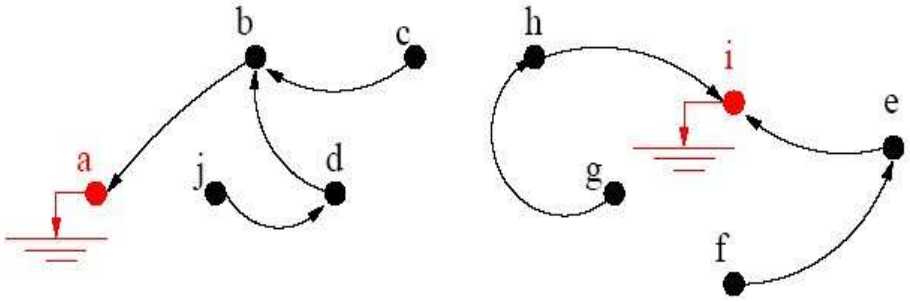


Figura 3.8: Exemplo de uma florestas espalhadas

Desta forma, podemos definir o que seriam os caminhos da floresta P, ou seja, para qualquer nó $t \in \mathcal{I}$, há um caminho $P^*(t)$ que é obtido percorrendo de frente para trás os nós predecessores ao longo do caminho. Por exemplo, na figura 3.8, o caminho $P^*(c) = < a, b, c >$, onde $P(c) = b,\ P(b) = a,$ e $P(a) = nil$; e o caminho $P^*(i) = < i >$, onde $P(i) = nil$.

3.2.5 Floresta de caminhos mínimos

Uma floresta de caminhos mínimos é uma floresta espalhada P, onde $f(P^*(t))$ é mínimo para todos os nós $t \in \mathcal{I}$ [2]. Considerando a função de custos f_{sum}, a figura 3.9 apresenta uma exemplificação:

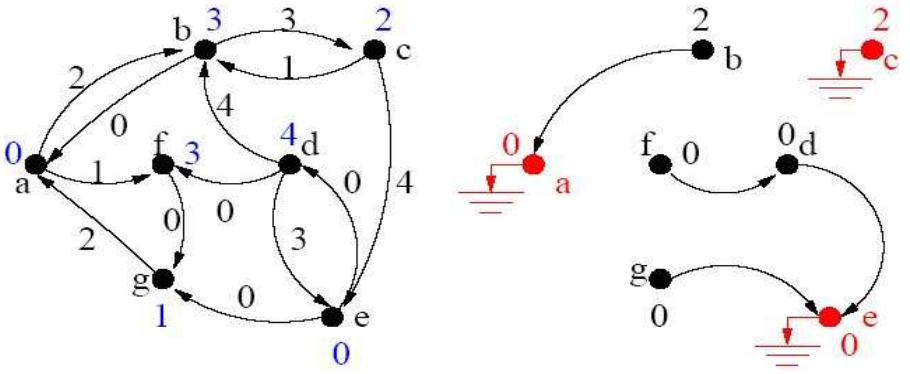


Figura 3.9: Exemplo de floresta de caminhos mínimos

Dentre os algoritmos clássicos para se encontrar um caminho mínimo em um grafo, destacamos neste trabalho o algoritmo de Dijkstra [35, 53, 56], o qual foi empregado pelos autores da IFT para sua implementação, como será abordado no capítulo seguinte.

O algoritmo de Dijkstra pode ser ligeiramente modificado para computar florestas de caminhos ótimos para funções de custo MI. O algoritmo também trabalha com funções de custo não-MI sob condições menores que sejam aplicadas somente a caminhos ótimos.

3.2.5.1 Algoritmo de Dijkstra: Entre um dado ponto e os demais

No algoritmo de Dijkstra [35], o objetivo é obter o menor caminho entre um dado vértice fixo e todos os demais vértices do grafo. Por exemplo, saber a distância mínima de São Paulo para todas as cidades do Estado. O algoritmo consiste basicamente em fazer uma visita por todos os nós do grafo, iniciando no nó fixo dado e encontrando sucessivamente o nó mais próximo, o segundo mais próximo, o terceiro mais próximo e assim por diante, um por vez, até que todos os nós do grafo tenham sido visitados.

Definindo melhor, escolhido um vértice como raiz da busca, este algoritmo calcula o custo mínimo deste vértice para todos os demais vértices do grafo. É considerado um algoritmo simples e com um bom nível de performance. Ele não garante, contudo, a exatidão da solução caso haja a presença de arestas com valores negativos.

Este algoritmo parte de uma estimativa inicial para o custo mínimo e vai sucessivamente ajustando esta estimativa. Ele considera que um vértice estará fechado quando já tiver sido obtido um caminho de custo mínimo do vértice tomado como raiz da busca até ele. Caso contrário considera-se o algoritmo aberto.

Quando todos os vértices tiverem sido fechados, os valores obtidos serão os custos mínimos dos caminhos que partem do vértice tomado como raiz da busca até os demais vértices do grafo. O caminho propriamente dito é obtido a partir dos vértices chamados de precedentes.

3.2.5.2 Algoritmo

Suponha que sejam dados **G** um grafo dirigido com peso nas arestas representado por lista de adjacências ou matriz de adjacências, e um vértice inicial $s \in V(\mathbf{G})$.

Vamos assumir a de dois vetores, uma estrutura para representar conjuntos indexados por V e uma fila de prioridades para os vértices que comporão o caminho[56]:

- S: num determinado ponto da execução do algoritmo S identifica os vértices que já têm distância determinada;
- d[ ]: num determinado ponto da execução do algoritmo d[v] armazena a distância de s para v se $v \in S$, ou $d[v] = \infty$ ou ainda d[v] é o comprimento de algum $s \rightarrow v^{-caminho}$ se $v \in V - S$ e o predecessor está em S:
- $\pi[]$: $\pi[v]$ é o predecessor de v no $s \rightarrow v^{-caminho}$

- Q: fila de prioridades com elementos de V - S e chave d[ ];
- Extrai-min(Q): remove de Q o elemento de prioridade maior prioridade que corresponde ao de menor chave.

Algoritmo 1 Algoritmo de Dijkstra

1. **Para** $(v = 1;\ v \leq |V|;\ v++)$ **faça**
2. $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow NIL$
4. **fim-Para**
5. $d[s] \leftarrow 0$
6. $S \leftarrow \emptyset$
7. $Q \leftarrow V$
8. **Enquanto** $(Q \neq \emptyset)$ **faça**
9. $u = \text{EXTRA-MIN}(Q)$
10. $S \leftarrow S \cup \{u\}$
11. **Para** $(T \in adj[u])$ **faça**
12. **Se** $(d[t] > d[u]) + \rho(u, t)$ **então**
13. $d[t] \leftarrow d[u] + \rho(u.t)$
14. $\pi[t] \leftarrow u$
15. **fim-Se**
16. **fim-Para**
17. **fim-Enquanto**

O algoritmo funciona da seguinte forma. No início $S = \{s\}$. Num determinado momento da execução antes da linha 7 no algoritmo descrito a seguir, a fila de prioridade Q contém os vértices de $V - S \neq \emptyset$ e S contém os vértices cujas distâncias já estão determinadas. Nesse ponto, retiramos o vértice u de maior prioridade de Q. Pela construção de Q esse é o vértice de $V - S$ mais próximo de s e portanto sua distância está determinada, u é incluído em S e o algoritmo prossegue, até que $Q = \emptyset$

Uma observação importante é que, na sua forma original, o algoritmo de Dijkstra computa apenas um único caminho de custo mínimo entre um dado par de vértices. Para se obter todos os caminhos de custo mínimo entre dois vértices é necessário modificar a forma de anotação dos precessores.

3.2.6 Funções de custo suaves

Uma função de custo f é **suave** se para qualquer nó $t \in \mathcal{I}$, há um caminho ótimo π terminado em t que pode ser trivial, ou possui a forma $\tau \cdot < s, t >$, onde [2]:

1. $f(\tau) \leq f(\pi)$,

2. τ é ótimo,

3. para qualquer caminho ótimo τ' terminado em s, $f(\tau' \cdot < s,t >) = f(\pi)$

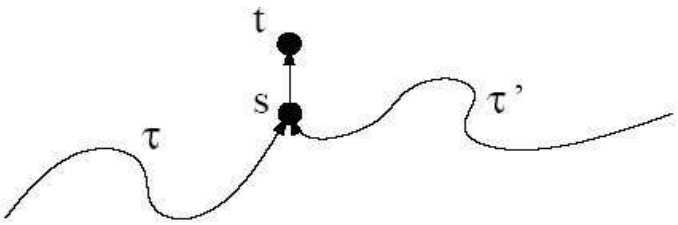


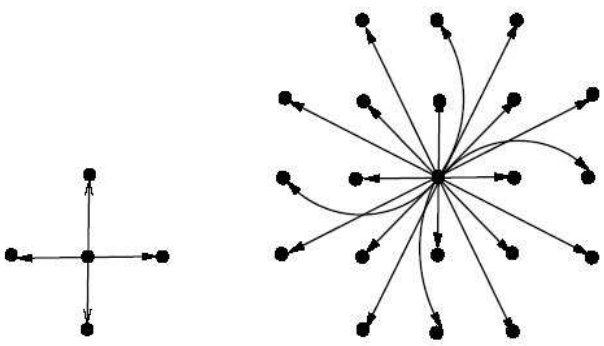
Figura 3.10: Representação da aplicação de funções de custo suaves

Essas condições são necessárias apenas para caminhos ótimos, diferentes da literatura anterior de grafos onde todos os caminhos devem satisfazer um conjunto de condições para garantir a corretude do algoritmo de Dijkstra.

3.3 Imagem como um grafo

Uma imagem $\hat{I}$ é um par $(\mathcal{I}, I)$ que consiste de um conjunto finito de pixels $\mathcal{I}$ (pontos em $\mathbb{Z}^2$) e um mapa I que atribui a cada pixel t em $\mathcal{I}$ um valor de pixel $I(t)$ em alguns valores arbitrários no espaço.

Uma relação de adjacência $\mathcal{A}$ é uma relação binária não-reflexiva entre pixels de $\hat{I}$, que é geralmente invariante a translação. Uma vez que a relação de adjacência $\mathcal{A}$ foi fixada, a imagem $\hat{I}$ pode ser interpretada como um grafo direcionado cujos nós são os pixels da imagem em $\mathcal{I}$ e cujos arcos são definidos por $\mathcal{A}$. Para este trabalho, utilizamos a **relação euclideana**, na qual um pixel $t = (x_t, y_t)$ é adjacente a um pixel $s = (x_s, y_s)$ (i.e. o arco $(s,t) \in \mathcal{A}$) se $(x_t - x_s)^2 + (y_t - y_s)^2 \leq \rho$.



(a) $\rho = 1$ (b) $\rho = 25$

3.3.1 Relação de Conectividade

Um pixel t é dito conectado ao pixel s se há um caminho de s para t no grafo [53]. O custo de um caminho no grafo é determinado por uma função de custo suave específica para cada aplicação, que geralmente depende de propriedades locais da imagem ao longo do caminho, tais como brilho, gradiente e a posição do pixel [56].

Capítulo 4

Transformada Imagem-Floresta (Image-Foresting Transform - IFT)

Muitos operadores da imagem podem ser direta ou indiretamente relacionados a uma partição da imagem em zonas de influência associadas com os pixels raízes, onde a zona de cada raiz consiste dos pixels que são conectados mais próximos a essa raiz do que a qualquer outra, em algum sentido apropriado.

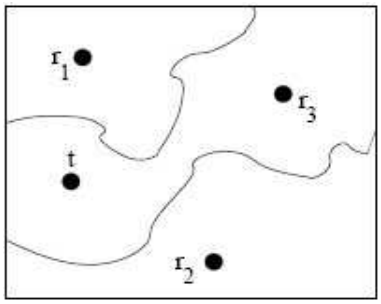


Figura 4.1: Pixel t está conectado mais próximo a raiz $r3$

O IFT unifica estes operadores de imagem computando uma floresta de caminhos ótimos em um grafo dirigido derivado da imagem. As raízes da floresta são extraídas de um conjunto dado de pixels sementes, que podem também consistir em todos os pixels da imagem. A zona da influência de uma raiz consiste nos pixels alcançados por um caminho de custo mínimo, considerando o custo de todos os caminhos no grafo a partir do conjunto de sementes para aqueles pixels.

Neste capítulo, procuramos apresentar o estudo e análise da Transformada Imagem-Floresta ou IFT. O método baseia-se nas relações de adjacências entre pixels, definindo um grafo na imagem, onde os pixels de uma imagem $\hat{I} = (\mathcal{I}, I)$ são os vértices deste grafo e, $(p, q) \in \mathcal{I}$ é uma aresta entre pixels adjacentes e, um pixel q é conexo a um pixel p se existir um caminho de p a q composto de pixels adjacentes no grafo.

A Transformada Imagem-Floresta [2] explora esta representação para reduzir problemas de processamento de imagens baseados em conectividade em um problema de floresta de caminhos de custo mínimo (caminhos ótimos). Portanto, para entendermos o trabalho realizado pela IFT na imagem, necessita-se de alguns conceitos básicos sobre imagens e grafos vistos nos capítulos anteriores.

O custo mínimo de um caminho é calculado por uma função dependente da aplicação que se deseja aplicar a IFT e com base em propriedades locais da imagem - tais como brilho, gradiente e posição de pixel ao longo do caminho. Para uma função de custo adequada, uma relação de adjacência e um dado conjunto de pixels sementes, a IFT associa a cada pixel da imagem um caminho de custo mínimo, particionando a imagem em uma floresta de caminhos ótimos onde cada árvore tem como raiz um pixel semente e, como nós os pixels da imagem mais "conexos" com a raiz do que com qualquer outra semente, em algum sentido apropriado. A IFT gera como resultado uma imagem anotada, onde cada pixel tem associado o predecessor no caminho ótimo, o custo deste caminho e o pixel raiz.

A principal referência das seções subseqüentes deste capítulo foi o trabalho de Falcão, Lotufo e Stolfi [2], que apresenta a teoria e os conceitos envolvidos na IFT.

4.1 Notação

Uma imagem $\mathbf{I}$ é um par $(\mathcal{I}, I)$ que consiste de um conjunto finito de pixels $\mathcal{I}$ (pontos em $\mathbb{Z}^2$) e um mapa I que atribui a cada pixel t em $\mathcal{I}$ um valor de pixel $I(t)$ em algum espaço de valores. Uma relação de adjacência $\mathcal{A}$ é uma relação binária irreflexiva entre pixels de $\mathcal{I}$, como foi definido no capítulo anterior.

Logo, um caminho é uma seqüência de pixels $\pi =< t_1, t_2, \ldots, t_k >$ onde (t_i, t_{i+1}) pertence a $\mathcal{A}$ para $1 \leq i \leq k-1$. Denotamos a origem t_1 e o destino t_k de π, por $org(\pi)$ e $dst(\pi)$, respectivamente. O caminho é trivial se $k = 1$. Se π e τ são caminhos tais que $dst(\pi) = org(\tau) = t$, denotamos por $\pi \cdot \tau$ a concatenação dos dois caminhos.

4.2 Definição

A IFT trabalha com uma imagem $\mathbf{I}$, uma função de custos f e uma relação de adjacência $\mathcal{A}$ e gera uma floresta de caminhos ótimos - uma floresta espalhada P tal que $P^*(t)$ é ótimo, para cada pixel t.

Note que, em uma floresta de caminhos ótimos P para uma função de custos restrita f^S a sementes, qualquer pixel t com custo infinito $f^s(P^*(t))$ pertencerá a uma árvore cuja raiz é um pixel semente.

A figura 4.2 ilustra o funcionamento da IFT. Podemos observar a representação de um grafo oriundo de uma imagem, com vizinhança-4 (adjacência), onde os números inteiros são os valores da imagem $I(t)$ (figura 5.1b). A seguir, é gerada a floresta de caminhos ótimos para a função de custos de caminho f_{max}, onde $h(t) = w(s,t) = I(t)$, restrita a três pixels sementes representados pelos pontos maiores (figura 5.1c).

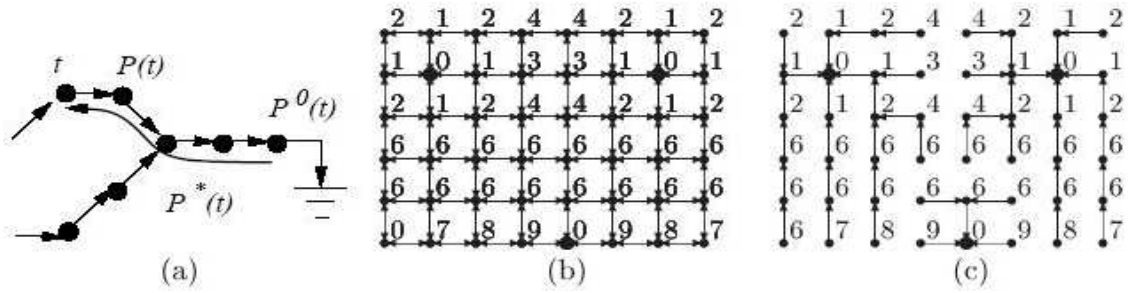


Figura 4.2: (a) Elementos principais numa floresta espalhada. (b) Um grafo de uma imagem com adjacência 4-conexa. (c) Uma floresta de caminhos ótimos para uma função de custo de caminho f_{max}. [2]

Em geral, podem existir muitos caminhos de custo mínimo que conduzem a um dado pixel, e muitas florestas de caminhos ótimos, porém, somente os custos de pixel $I(t)$ são definidos de modo único. Observamos que, se tomarmos independentemente um caminho ótimo para cada pixel, a união desses caminhos pode não ser uma floresta. Provavelmente, certos grafos e funções de custos podem não admitir qualquer floresta de caminhos mínimos. As condições suficientes para a existência da IFT são apresentadas nas seções seguintes.

4.3 Funções de Custo Mínimo

A solução de cada problema utilizando IFT requer a escolha de uma função f específica, que associa um custo, que provêem de um conjunto V ordenado de valores com valor máximo denotado por $+\infty$, para cada caminho. Para garantir uma floresta de caminhos ótimos, esta função deve ser suave, segundo as condições definidas no capítulo anterior.

Um exemplo mais comum é a função de custo aditiva (soma dos custos do caminho

entre dois nós), a qual satisfaz:

$$f_{sum}(< q >) = h(q), \tag{4.1}$$
$$f_{sum}(\pi \cdot < p, q >) = f_{max}(\pi) + w(p, q)$$

Onde $(p, q) \in A$, π é qualquer caminho terminado em p, $h(q)$ é um custo inicial fixo para qualquer caminho iniciado em q, e $w(p, q)$ é um peso não negativo associado ao arco *(p,q)*.

4.4 Pixels Sementes

O conjunto $S \subseteq \mathcal{I}$ de pixels sementes restringem a busca por caminhos ótimos que iniciem em S. Isto equivale a modificar a função f de custo para f^S, a qual satisfaz

$$f^S(\pi) = \begin{cases} f(\pi), \ se \quad org(\pi) \in S, \\ +\infty, \ \text{no caso contrário} \end{cases} \tag{4.2}$$

Todas as raízes da IFT são pixels sementes, mas nem todas sementes se transformam em raízes da IFT, pois o custo de um caminho trivial $< q >$, onde $q \in S$, pode ser maior que o custo de um outro caminho iniciado em S com término em q.

4.5 Imagem Anotada

Um mapa P de predecessores é uma função que associa a cada pixel $q \in \mathcal{I}$, ou outro pixel $p \in \mathcal{I}$, $(p, q) \in \mathcal{A}$, ou uma marca $nil \neq \mathcal{I}$. No segundo caso, q é dito ser uma raiz do mapa. Uma floresta espalhada é um mapa de predecessores que não contém ciclos, isto é, aquele que leva todo pixel para nil em um número finito de iterações.

Para qualquer pixel $q \in \mathcal{I}$, a floresta P define um caminho $P^*(q)$ recursivamente como:

$$\begin{cases} < q >, \quad se \quad P(q) = nil, \quad \text{e} \\ P^*(q) \cdot < p, q > \quad se \quad P(q) = p \neq nil \end{cases}$$

A IFT calcula essencialmente uma floresta P de caminhos ótimos, isto é, uma floresta espalhada, onde $P^*(q)$ tem custo mínimo para todo $q \in \mathcal{I}$. Para fins de eficiência, a IFT também gera um mapa de custos C e um mapa de raízes L, onde $C(q)$ é o custo do

caminho ótimo até q e $L(q)$ é o pixel inicial deste caminho.

4.6 Algoritmos e Estrutura de Dados para IFT

O algoritmo geral da IFT é essencialmente o Algoritmo de Dijkstra estendido para múltiplas fontes e funções de custo de caminho suaves. Note que, embora possam existir várias florestas P de caminhos de custo mínimo que satisfazem um dado problema, o mapa C de custos ótimos deve ser único. Esta ambigüidade é parcialmente resolvida quando decidimos pelo caminho de menor custo que encontra um dado pixel primeiro. O algoritmo resultante é apresentado a seguir:

Entrada: Imagem $\hat{I} = (\mathcal{I}, I)$, relação de adjacência $\mathcal{A}$, e função f de custo de caminho suave. Saída: Imagens $C' = (\mathcal{I}, C)$ de custo, $P' = (\mathcal{I}, P)$ de predecessores, e $L' = (\mathcal{I}, L)$ de raízes. Auxiliares: Fila Q de prioridade e variável c.

Algoritmo 2 Algoritmo IFT1

1. **Para todo** pixel $p \in \mathcal{I}$ **faça**
2. $C(p) \leftarrow f(< q >),\ P(p) \leftarrow nil,\ L(p) \leftarrow p$
3. **Se** $C(p) < +\infty$ **então**
4. Insira p *em* Q
5. **fim-Se**
6. **fim-Para**
7. **Enquanto** $(Q \neq \emptyset)$ **faça**
8. Remova um pixel p *de* Q cujo custo $C(p)$ é mínimo
9. **Para todo** $q \in \mathcal{A}(p),\ tal\ que\ C(q) > C(p)$ **faça**
10. $c \leftarrow f(P^*(q) \cdot < p, q >)$
11. **Se** $C(q) \neq +\infty$ **então**
12. Remova q *de* Q
13. **fim-Se**
14. $C(q) \leftarrow c,\ P(q) \leftarrow p,\ L(q) \leftarrow L(p),$ e insira q *em* Q
15. **fim-Para**
16. **fim-Enquanto**

Aplicando a regra de desempate acima, inclusive para pixels distintos que são encontrados por caminhos ótimos de mesmo custo, aquele que entrou na fila Q primeiro é o primeiro a sair. Neste caso dizemos que a fila Q segue a política *First-In-First-Out* (FIFO) de desempate. Outra forma de resolver parcialmente o problema de ambigüidade

das florestas ótimas é implementar a fila Q com política *Last-In-First-Out* (LIFO) de desempate. Este variante é apresentado a seguir:

Entrada: Imagem $\hat{I} = (\mathcal{I}, I)$, relação de adjacência $\mathcal{A}$, e função f de custo de caminho suave. Saída: Imagens $C' = (\mathcal{I}, C)$ de custo, $P' = (\mathcal{I}, P)$ de predecessores, e $L' = (\mathcal{I}, L)$ de raízes. Auxiliares: Fila Q de prioridade e variável c.

Algoritmo 3 Algoritmo IFT2

1. **Para todo** pixel $p \in \mathcal{I}$ **faça**
2. $\quad C(p) \leftarrow f(< q >),\ P(p) \leftarrow nil,\ L(p) \leftarrow p,$ *e insira* p *em* Q
3. **fim-Para**
4. **Enquanto** $(Q \neq \emptyset)$ **faça**
5. $\quad$ Remova um pixel p *de* Q cujo custo $C(p)$ é mínimo
6. $\quad$ **Para todo** $q \in \mathcal{A}(p),$ *tal que* $q \in Q$ **faça**
7. $\quad\quad c \leftarrow f(P^*(q) \cdot < p, q >)$
8. $\quad\quad$ **Se** $c \leq C(q)$ **então**
9. $\quad\quad\quad$ Remova q *de* Q, faça $C(q) \leftarrow c,\ P(q) \leftarrow p,\ L(q) \leftarrow L(p),$ e insira q *em* Q
10. $\quad\quad$ **fim-Se**
11. $\quad$ **fim-Para**
12. **fim-Enquanto**

A principal diferença entre este último algoritmo e o anterior está na linha 9, onde $P(q)$ e $L(q)$ devem ser atualizados, e q deve ser removido e reinserido em Q, mesmo quando c é igual a C*(q)*. Com a política FIFO, qualquer conjunto conexo de pixels que poderia ser encontrado a partir de duas ou mais raízes por caminhos ótimos de mesmo custo será particionado entre as respectivas árvores. No caso da política LIFO, esses pixels são associados a árvore cuja raiz foi a última a encontrar o conjunto. A política LIFO é útil em algumas situações, como no cálculo do número de mínimos regionais de uma imagem, mas a política FIFO satisfaz melhor as expectativas do usuário com relação à partição da imagem (por exemplo, na segmentação). Ambas não resolvem por completo o problema da ambigüidade das florestas ótimas e regras extras de desempate podem ser implementadas em Q.

4.6.1 Fila de Prioridade

Os algoritmos da IFT têm complexidade $O(m\ +\ n\ log\ n)$, onde $n = |\mathcal{I}|$ é o número de nós (pixels) e $m = |\mathcal{A}|$ o número de arcos, considerando a implementação para a fila de prioridade Q usando um *heap* binário [57].

A implementação mais simples para a fila Q utiliza-se um *heap* binário, como apresentado por Falcão, Lotufo e Stolfi [2]. Neste caso os algoritmos acima terão complexidade $O(m + n \log n)$, onde $n = |Np|$ é o número de nós (pixels) e $m = |A|$ o número de arestas.

Na maioria das aplicações, podemos usar funções de custo de caminho com incrementos de custo inteiros e limitados a uma constante K ao longo do caminho. Isto permite a utilização da fila circular de Dial [58] com $K+1$ posições. Dial [58] publicou em 1969 uma variante do algoritmo de Dijkstra [56, 58], para os casos especiais onde o custo dos arcos são inteiros no intervalo $[0..K]$, o qual usa a classificação por buckets para se alcançar um tempo de execução $O(m + nK)$. Cada posição i, $i = 0, 1, \ldots, K$, deve armazenar uma lista duplamente ligada de todos os pixels p com custo $i = C(p)$.

Podemos notar que a cada instante existe um valor mínimo C_{min} e um valor máximo C_{max} de custo para os pixels armazenados em Q. A diferença $C_{max} - C_{min} \leq K$ deve ser mantida para garantir a corretude da fila. Em algumas aplicações sabe-se que os incrementos são inteiros e limitados, mas não antes de inserir um novo pixel devemos verificar a necessidade de realocar ou não mais elementos para a fila.

Em muitas aplicações, os custos de caminho $f(\pi)$ são tanto $+\infty$ ou inteiros no intervalo $[C_{min}..C_{max}]$. Nestes casos, pode-se implementar a fila Q como um vetor de cubos C_Δ, cada um apontando para lista circular duplamente ligada de pixels, onde $C_\Delta = C_{max} - C_{min} + 1$. A inserção, deleção, e a atualização do custo de um pixel pode então ser feita no tempo $O(1)$. Encontrando o próximo pixel com custo mínimo em Q, pode-se saltar por vários cubos vazios; entretanto, desde que o custo do pixel seguinte nunca diminua, o tempo total para a etapa 8 do Algoritmo 1 é $O(C_\Delta)$. O Algoritmo 1 executará, então, em tempo $O(m + n + C_\Delta)$ e armazenamento de $O(n + C_\Delta)$. Podemos notar que cada pixel pode aparecer na fila mais de uma vez, e assim pode-se evitar a alocação dinâmica pré-alocando duas ligações $next(p)$ e $prev(p)$ para cada pixel p, que são usados para conectar os pixels pertencentes a cada cuba [59].

4.7 Aplicações

Podemos citar alguns exemplos de problemas redutíveis a uma IFT, como a segmentação por transformada de watershed, segmentação baseada em conectividade *fuzzy*; a filtragem e segmentação por reconstrução morfológica, segmentação por crescimento de regiões, segmentação por perseguição de borda, cálculo de caminhos geodésicos, transformadas de distância, representação por esqueletos multiescala, estimação de pontos de saliência em

curvas e cálculo da dimensão fractal multiescala de uma curva.

Alguns casos podem não ser facilmente relacionados com um problema de partição da imagem em uma floresta de caminhos ótimos (como reconstrução morfológica, perseguição de bordas, pontos de saliência). Porém, na maioria dos casos, a solução é obtida pela simples escolha de parâmetros da IFT seguida de um processamento local da imagem anotada, em tempo proporcional ao número de pixels. Este resultado é obtido com uma extensão do Algoritmo de Dijkstra para múltiplas fontes e função de custo de caminho mais geral.

Diversos operadores de imagem podem ser reduzidos a simples escolha de parâmetros da IFT e uma operação local sobre a imagem anotada. Nesta seção, vamos abordar apenas os operadores relacionados com segmentação e filtragem de imagens.

4.7.1 Transformada da Distância Euclideana - EDT

Considere como sementes em S os pixels da borda de um objeto (i.e. pixels do objeto que possuem a menos um vizinho-4 no fundo). Muitas aplicações requerem o cálculo da **distância mínima** entre pixels do objeto e/ou fundo e o conjunto

$$f_{euc}^{S}(< q >) = \begin{cases} 0, \text{ se } q \in S, \\ +\infty, \text{ no caso contrário} \end{cases} \tag{4.3}$$

$$f_{euc}^{S}(\pi \cdot < p, q >) = (x_{org(\pi)} - x_q)^2 + (y_{org(\pi)} - y_q)^2$$

onde $org(\pi)$ é o pixel inicial do caminho ótimo π. Neste caso, os caminhos ótimos não estão restritos ao domínio do objeto/fundo. Esta transformada de distância é dita **Euclideana** sempre que a escolha da relação de adjacência $\mathcal{A}$ for suficiente para que todos os pixels da imagem sejam alcançados pelo pixel semente mais próximo. Caso contrário, a função f_{euc}^{S} não é suave. Na prática, porém, a adjacência-8 é suficiente na maioria dos problemas.

Podemos também calcular outras transformadas de distância usando f_{sum} e $\delta(p, q)$ dado por uma das métricas acima, exceto a Euclideana.

Sendo assim, a *transformada da distância Euclideana* (EDT) de um dado conjunto de sementes S fornece para cada pixel da imagem t um valor $C(t)$ que é a distância mínima Euclideana de t para S. (De fato, a fim de se evitar números irracionais, o quadrado da

distância é armazenado preferivelmente).

A EDT é relacionada ao conceito do *Diagrama Discreto de Voronoi* (DVD), que é a partição dos pixels da imagem em *regiões discretas de Voronoi*. Maiores detalhes sobre EDT e as variações de sua implementação podem ser vistas em [2, 60, 61, 62]

4.7.2 Mínimos Regionais

Os mínimos regionais de uma imagem $\hat{I} = (\mathcal{I}, I)$ podem ser calculados diretamente do mapa L de raízes, com a função f_{ini} descrita abaixo:

$$f_{ini}(< q >) = I(q), \text{para todo } q \in \mathcal{I}$$

$$f_{ini}(\pi \cdot < p, q >) = \begin{cases} f_{ini}(\pi), \text{ se } I(p) \leq I(q), \\ +\infty, \text{ no caso contrário} \end{cases} \tag{4.4}$$

Os mínimos regionais podem ser usados como sementes da IFT em algumas tarefas de segmentação. Com a política de desempate FIFO um pixel será raiz da IFT se e somente se ele pertencer e um mínimo regional. Portanto, uma imagem binária dos mínimos regionais pode ser gerada associando 1 a pixels raízes e 0 aos demais. Com a política LIFO, vamos obter exatamente um pixel por mínimo regional. Neste caso, temos uma contagem direta do número de mínimos e a extensão desses mínimos na imagem é obtida podando as arvoras com raiz r para escolher os pixels p com $I(p) = I(r)$.

4.7.3 Transformada de Watershed

Considere uma imagem $\hat{I} = (\mathcal{I}, I)$ onde $I(p)$ é a altitude dos pixels. A transformada clássica de watershed simula a inundação desta superfície por fontes de água colocadas uma em cada mínimo regional; e uma barreira (linhas de watershed) sendo erguida toda vez que águas provenientes de fontes distintas se encontram, impedindo assim que elas se misturem. Essas linhas podem ser obtidas direto do mapa L de raízes (bacias), usando a função f_{peak} (caso particular de f_{max}) com $h(q) = I(q) + 1$ para todo pixel $q \in \mathcal{I}$ e $w(p, q) = I(q)$.

$$f_{peak}(< q >) = h(q),$$

$$f_{peak}(\pi \cdot < p, q >) = max\{f_{peak}(\pi), I(q)\}$$

onde $h(q) \leq I(q)$ (imposição de marcadores), se $q \in S$, e $h(q) = +\infty$ no caso contrário.

O custo de um caminho ótimo com término em um platô será seu prórpio brilho na imagem original. As bacias sem sementes se transformam em platôs na imagem $\hat{C}$ de custos com valores iguais à altura da água que as encontram. As raízes da floresta serão os pixels sementes, que estarão em mínimos regionais da imagem de custos. Em outras palavras, se calculássemos uma transformada de watershed na imagem de custos usando seus mínimos regionais como sementes, obteríamos o mesmo resultado (i.e. não precisamos de mudanças de homotopia).

Note que as linhas de watershed podem ser obtidas da imagem $\hat{R}$ de raízes. Para obter linhas de watershed com espessura máxima de 2 pixels, classificamos como pertencentes à linha todos os pixels p com raiz $R(p) \neq R(q)$ para algum q vizinho-4 de p. Se atribuímos um número inteiro $\lambda(q), q \in S$, distinto para cada fonte q, podemos obter linhas com espessura de 1 pixel, basta classificarmos como linha todos os pixels p com $\lambda(R(p)) < \lambda(R(q))$ para algum q vizinho-4 de p.

Podemos gerar uma imagem cinza, onde o brilho das linhas é a menor altura entre as respectivas zonas de influência de suas raízes, e o brilho dos demais pixels é zero. A limiarização desta imagem gera uma família de imagens de bordas.

A detecção automática de sementes é a tarefa mais complicada. Exemplos comuns são usar como sementes: pixels obtidos por limiarização, mínimos/máximos regionais e pixels da borda da imagem. Esses mínimos, por exemplo, podem ser calculados com uma IFT de função f_{ini} mostrada na subseção anterior.

Existem vários possíveis variantes para uma transformada de watershed usando a IFT. Maiores detalhes a respeito podem ser encontrados em [2], onde também são apresentados resultados interessantes destas variações.

4.8 IFT para Perseguição de bordas de objetos

Quando examinamos a intensidade dos pixels em torno da borda de um objeto em uma imagem cinza percebemos que existe uma incerteza com relação à posição exata da borda (abordagem *hard*) [35]. Poderíamos transferir nossa incerteza para um conjunto de pixels que forma uma faixa de largura variável em torno da borda e usar este conjunto de pixels para representá-la (abordagem *fuzzy*) [29].

Uma borda de objeto em uma imagem é caracterizada por uma descontinuidade de propriedades da imagem, internas e externas ao objeto em uma vizinhança da borda.

Portanto, uma função de custo de caminho para perseguição de bordas deve capturar esta descontinuidade associando **custos baixos** para pixels adjacentes sobre a borda desejada, e custos altos no caso contrário. Dados dois pixels, o como origem e d como destino, sobre a borda, o custo do caminho ótimo de o a d deve ser um segmento de borda. Na prática, porém, uma borda consiste de vários segmentos ótimos formando um contorno fechado.

Podemos considerar k conjuntos S_i, $i = 1, 2, \ldots, k$, de pixels sementes selecionados sobre uma borda, tais que $S_1 = S_k$ possui um único pixel e os pixels em S_i, $i = 2, 3, \ldots, k-1$, pertencem a uma região pequena de incerteza (e.g. uma linha que cruza a borda, uma marca circular sobre a borda). Um segmento ótimo de borda de S_i para S_{i+1}, $i = 1, 2, \ldots, k-1$, pode ser formado por n caminhos de custo mínimo que atingem todos os pixels de S_{i+1}. Neste caso, teríamos um segmento ótimo de **borda fuzzy**, ou um único caminho de menor custo como segmento ótimo de borda *hard*. Para simplificar, vamos considerar apenas um único caminho ótimo 4- ou 8-conexo entre cada par S_i e S_{i+1}, ou seja apenas o caso de contornos ótimos.

Portanto, uma borda é um **contorno ótimo** que passa pela seqüência de conjuntos S_i, $i = 1, 2, \ldots, k$, de pixels sementes. Isto requer $k-1$ IFTs para calcular os caminhos de S_i a S_{i+1}, $i = 1, 2, \ldots, k-1$.

A escolha da função de custo deve levar em conta dois aspectos relevantes:

1. Bordas próximas com o mesmo grau de descontinuidade devem ser evitadas com pré-processamento ou explorando propriedades, tais como a orientação da descontinuidade. Por exemplo, custos baixos podem ser atribuídos para arcos de borda no grafo, cujo o lado esquerdo tem brilho menor/maior que o brilho do lado direito.

2. Na ausência de informação de borda, custos altos podem ser atribuídos a alguns arcos da borda, e portanto, a função de custo deve ser **robusta** com relação a estes casos. Por exemplo, a função f_{max} **não é adequada**, pois basta um arco com custo máximo para o conjunto de pixels de destino possa ser atingido por diversos caminhos ótimos, passando por outras regiões da imagem longe da borda.

Esta reflexão nos leva a funções de custo do tipo f_{sum} com política FIFO, tais como

f_{ctrack} abaixo:

$$F_{ctrack}(< q >) = \begin{cases} 0, & q \in S \\ +\infty, \text{no caso contrário} \end{cases}$$

$$F_{ctrack}(\pi \cdot < p, q >) = F_{ctrack}(\pi) + (K - Max\{G(p,q) \cdot \eta(p,q), 0\}) \tag{4.5}$$

onde $h(q) = 0$, se $q \in S_1$ na primeira interação, ou $h(q)$ é o custo final do pixel q na iteração anterior, se $q \in S_i$ e $h(q) = +\infty$ no caso contrário, independente da iteração; $G(p,q)$ é um vetor gradiente estimado no ponto médio do arco (p,q); $\eta(p,q)$ é o arco (p,q) rotacionado de 90 graus no sentido anti-horário; e K é um limite superior para $|G(p,q) \cdot \eta(p,q)|$. O vetor $G(p,q)$ deve ser tal que arcos sobre a borda orientada do objeto possuam valores baixos de custo e os demais arcos fiquem com valores altos de custo. Note que a cada iteração, o algoritmo pode parar o cálculo da IFT quando o último pixel de S_{i+1} sai da fila Q. O contorno final pode ser obtido das respectivas $k-1$ imagens de predecessores $\hat{P}_{k-1}, \ldots, \hat{P}_1$, mas também é possível modificar o algoritmo da IFT para usar um único conjunto de imagens de custos e de predecessores. Antes de iniciar uma iteração i, os segmentos ótimos que atingiram os conjuntos S_j, $j \leq i$, nas iterações anteriores são candidatos a fazerem parte do contorno ótimo final. Portanto, este segmentos devem permanecer com os custos e os predecessores calculados, enquanto os demais pixels da imagem devem ter seus custos reinicializados para $+\infty$. Um cuidado especial, porém, deve ser tomado na última iteração $k-1$, pois o pixel em S_1 também deve ter seu custo reinicializado para $+\infty$ e o algoritmo pára quando ele sai da fila Q. O contorno ótimo é obtido percorrendo os predecessores deste pixel até encontra-lo novamente.

Na figura 4.3, vemos a borda traçada pela IFT de um determinado objeto. A entrada da IFT é uma imagem colorida no formato PPM, e no final do processo, a imagem resultante é gerada no formato PGM, ou seja, em tons de cinza.

Figura 4.3: Borda traçada pela IFT num objeto

4.8.1 O método *live-wire*

O método *live-wire* [12] é um exemplo que utiliza esta abordagem de forma interativa. Para segmentar uma borda, o usuário seleciona a primeira semente (pixel em S_1) e o método calcula uma IFT em toda a imagem, cuja única árvore terá como raiz a semente selecionada. Para cada posição subseqüente do cursor, o algoritmo mostra na tela o caminho de custo mínimo da raiz até esta posição. O usuário pode mover livremente o cursor sobre a imagem e verificar os caminhos ótimos. Quando o cursor se aproxima da borda desejada, este caminho gruda na borda e o usuário então seleciona a posição atual do cursor (pixel em S_2) para confirmar o segmento ótimo da borda. O processo se repete a partir do pixel selecionado até o usuário decidir fechar o contorno.

O algoritmo da IFT também pode ser modificado para ser executado de forma incremental. O método *live-wire-on-the-fly* usa esta variante. Maiores detalhes pode ser encontrado no trabalho de Falcão, Udupa e Miyazawa [59].

Capítulo 5

Descrição da aplicação

Como vimos no capítulo anterior, para perseguição de bordas, para alguns tipos de imagens os resultados não são satisfatórios, mas que podem ser resolvidos como veremos neste capítulo. Para isso descreveremos, para um conjunto de imagens de interesse, os problemas encontrados na utilização da Transformada Imagem-Floresta, IFT.

O objetivo deste trabalho é identificar um ou vários objetos através de sua borda, utilizando a IFT como método escolhido para tal processo, tendo como amostra para nossos testes, o conjunto de imagens das figuras 5.1, 5.2, 5.3, 5.4, 5.5.

Figura 5.1: Imagem 01

Figura 5.2: Imagem 02

Figura 5.3: Imagem 03

Figura 5.4: Imagem 04

Figura 5.5: Imagem 05

As imagens contém fragmentos de artefatos arqueológicos [63] sobre um fundo quadriculado, além de outros elementos (como sombras, gabaritos), onde a identificação destes fragmentos será importante em pesquisas futuras. Deseja-se obter, portanto, o contorno de cada fragmento com o máximo de precisão possível.

Ao aplicarmos a IFT numa das imagens (por exemplo, a figura 5.3), temos o seguinte resultado com alguns objetos identificados:

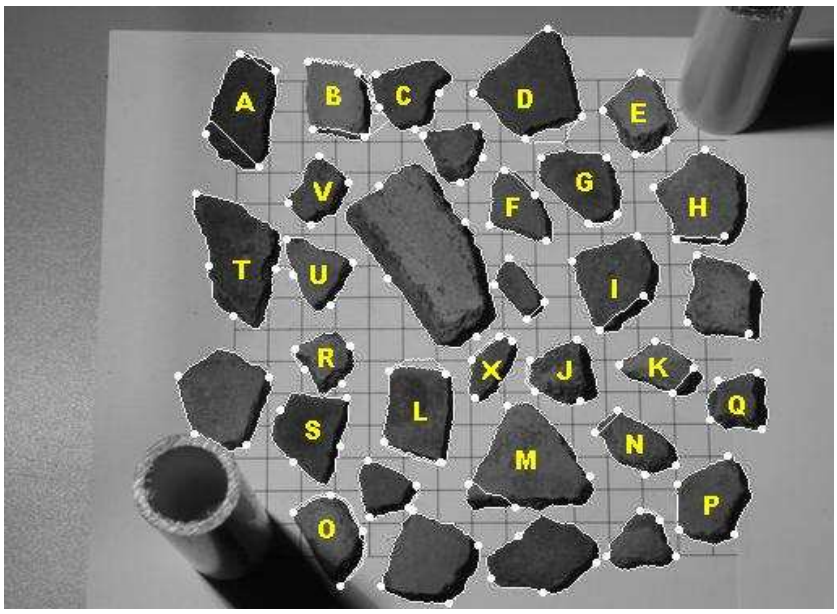


Figura 5.6: Detecção de bordas por IFT

Estes resultados apresentados pela figura 5.6 foi obtido utilizando-se a forma padrão da IFT, ou seja, orientação para borda pela imagem de entrada, escolha de pixels sementes sem preocupação com a orientação e posição e a função de custos f_{ctrack} descrita no capítulo 4. Nesta imagem observamos que ao se aplicar a técnica da IFT para detecção de contornos dos objetos de interesse, apresentou os seguintes problemas:

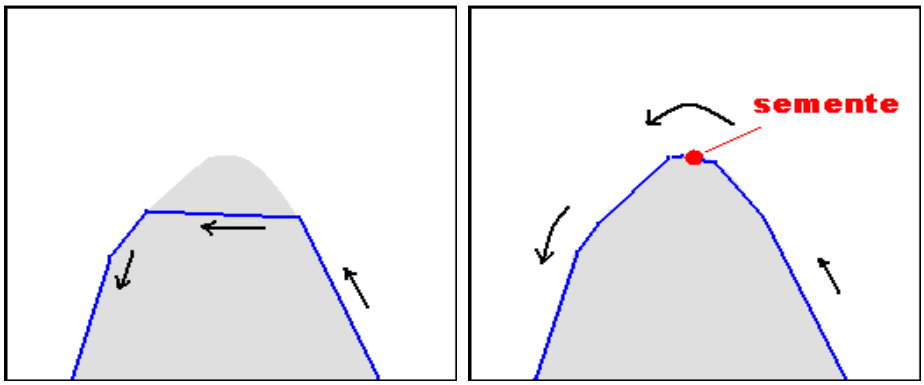
- Escolha dos pixels sementes
- Ordenação dos pixels sementes, gerando arestas duplas
- Arestas espúrias

Descreveremos com mais detalhes estes problemas a seguir, abordando também as devidas soluções adotadas.

5.1 Problema da escolha dos pixels sementes

Em alguns objetos de interesse onde traçamos seu contorno pela IFT, o traçado era feito de forma irregular, como vemos, por exemplo, nos objetos M e K da figura 5.6. Neste caso, a perseguição de bordas é afetada pela posição dos pixels sementes.

No decorrer dos testes, vimos que normalmente precisamos de um pixel semente para cada saliência mais evidente da borda do objeto, ou seja, em pontos onde há uma mudança brusca da borda do objeto (pontos de alta curvatura ou corners [64]). A função da IFT para perseguição de bordas possui nestes casos um comportamento irregular, ilustrado pela figura 5.7. A IFT em pontos de alta curvatura não consegue realizar o contorno nesta região, ultrapassando a imagem para seguir seu traçado. Desta forma, ao colocarmos um pixel semente nesta região, o traçado passa por esta semente e, portanto, encontrando o correto contorno desejado.



(a) Traçado incorreto

(b) Traçado correto com acréscimo de um pixel semente no ponto de alta curvatura

Figura 5.7: Pixels sementes em pontos de alta curvatura de borda

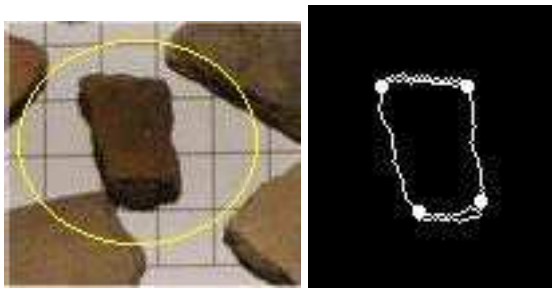
Assim, ao posicionar os pixels sementes nos pontos citados acima, ocorre uma melhora na eficiência da IFT, ou seja, o contorno é traçado corretamente como podemos observar, por exemplo, nos objetos P, Q, S e V da figura 5.6.

Como selecionamos manualmente os pixels sementes sobre a borda do objeto - através da ampliação da imagem num tamanho suficiente para se ter exatidão dos pixels desejados - o processo desta escolha, portanto, não é automatizado, acarretando lentidão no processo global de traçado do contorno.

5.2 Problema da ordenação de sementes

Na experimentação da técnica realizada nos estudos, observamos que o traçado é feito de maneira eficiente, porém, a escolha dos pixels sementes é importante e torna-se uma das dificuldades no uso da IFT. Esta escolha depende muito da característica do objeto cujo contorno de sua borda desejamos traçar. Algumas questões relevantes sobre o caso das sementes, tais como quantas sementes são necessárias e, se há alguma ordenação a ser estipulada entre essas sementes para realizar o percurso foram identificadas. Contudo, podemos intuir que, como elucidado na seção 5.1 anterior, se precisamos de um pixel semente para cada ponto de alta curvatura na borda do objeto, o número de pixels sementes que necessitamos a um determinado objeto será igual ao número de pixels selecionados nestes pontos de alta curvatura.

Resolvido a escolha do número de pixels sementes experimentalmente, procuramos determinar uma ordem para as sementes onde a borda deve passar obrigatoriamente. Em alguns testes como na figura 5.8, vimos uma aparência de linha duplicada no traçado (observado também nos objetos A, B, F, H, I, K, L, N e X da figura 5.6). Isto se deve a ordem errada dos pontos, ou seja, o traçado vai até um ponto k, volta ao ponto $k-1$, e depois segue até o $k+1$ (Figura 5.8). Ao modificarmos a ordenação das sementes o problema se resolve, ou seja, desejamos a combinação da ordem das sementes onde o custo total do contorno é mínimo.



(a) Imagem-alvo (b) Borda isolada da imagem

Figura 5.8: Bordas com aspecto de linha duplicada

Desta forma, a IFT traça corretamente a borda do objeto, como podemos observar, por exemplo, nos objetos R, S, T e V da figura 5.6 como outros objetos não identificados, onde essa ordenação foi empregada. Porém, em objetos muito próximos, ocorre o aparecimento de arestas espúrias, problema este que será abordado na próxima seção.

5.3 Problema das arestas espúrias

Quando temos imagens com objetos muito próximos entre si, ocorre o aparecimento de arestas espúrias, observado na figura 5.6 entre os objetos B, C e T, U além de D. Para ilustrar o problema, vejamos a figura 5.9 a seguir, onde observa-se as arestas nas bordas nas regiões de maior proximidade:

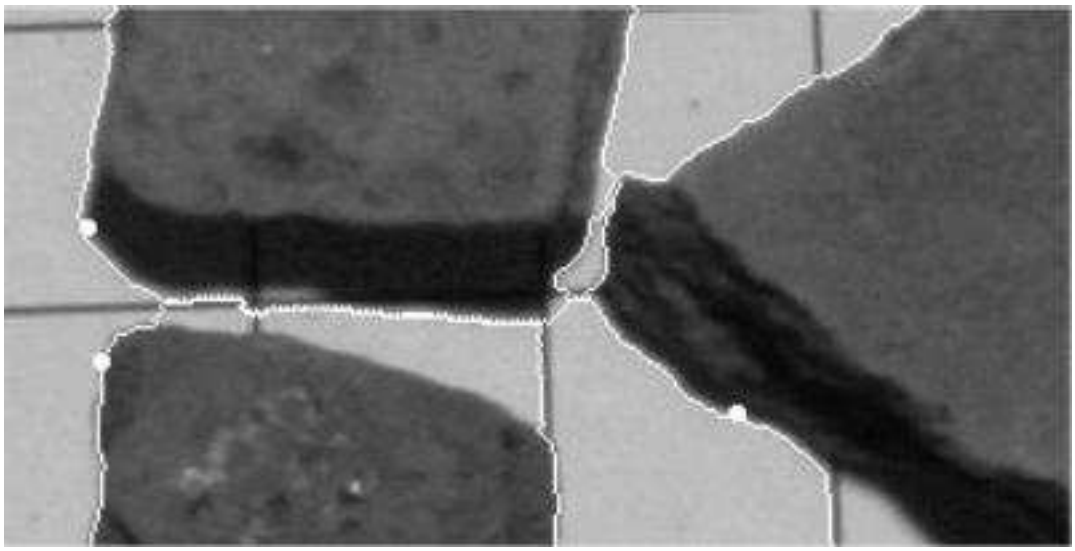


Figura 5.9: Ocorrência de arestas espúrias no traçado da IFT.

Ao detectarmos a borda de alguns objetos, podemos observar que, mesmo havendo uma pequena distância entre os mesmos, o traçado alcançado não os individualiza, ou seja, aparecem arestas do contorno que impossibilitam o traçado correto dos contornos dos objetos. Os contornos parecem se desprender dos objetos que os originam, se confundindo até com os contornos dos demais.

A figura 5.10 demonstra como o problema ocorre. Temos dois objetos muito próximos entre si em uma imagem, denominados O1 e O2. Deseja-se aplicar a IFT para detectar a borda do objeto O2:

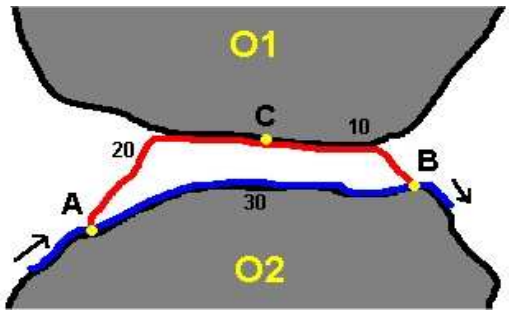


Figura 5.10: Representação da ocorrência de arestas espúrias no traçado da IFT

Conforme visto no capítulo 4, na figura 5.10 aparecem representados os custos obtidos pela função f_{ctrack} (seção 4.8). Podemos, então, observar que o traçado do caminho entre os pixels A e B parece sofrer um desvio em sua rota no objeto-alvo e isso se explica pelo comportamento da IFT em relação à busca do melhor caminho entre dois pontos na borda, próximo ao objeto vizinho.

Quando o traçado da IFT chega ao pixel A do objeto O2, a fim de continuar seu traçado correto, ele deveria prosseguir até o pixel B, entendendo que o custo de (A,B) seja igual (podendo também ser até menor) a 30. Porém, há um caminho de A para B passando pelo pixel C do objeto O2 que corresponde ao mesmo valor 30 de custo, e já que desejamos o menor caminho entre A e B, temos essas duas alternativas. Obviamente, a segunda possibilidade não é a ideal para o objetivo desejado, contudo, a IFT opta por este caminho, gerando, assim, uma aresta espúria.

A solução que propomos se baseia em não permitir que a IFT desvie seu traçado da borda do objeto, seguindo seu percurso através dos pixels que o compõem, mantendo a integridade de seu contorno, conforme apresentamos no próximo capítulo.

5.3.1 Solução para o problema das arestas espúrias

Para resolver o problema das arestas espúrias, propomos as seguintes etapas: pré-processamento nas imagens de entrada e mudança da função de custo. Além disso, apresentamos os resultados obtidos com a abordagem proposta e, por fim, realizamos uma discussão a respeito.

5.3.2 Pré-Processamento para eliminar as arestas espúrias

Denominamos pré-processamento a etapa intermediária do processo de resolução das arestas espúrias e conseqüente melhoria da detecção de bordas pela IFT. Esta etapa pode ter três passos de execução, e sua necessidade dependerá das imagens de entrada:

1. Realce dos objetos nas imagens de entrada.
2. Cálculo do gradiente.
3. Segmentação por limiarização.

5.3.2.1 Realce dos objetos nas imagens de entrada

Podemos observar que nestas imagens há linhas formando um aspecto de grade no plano de fundo dos fragmentos, bem como a formação de sombras causadas pela posição da luz na obtenção das imagens. Para que estas não atrapalhassem o processo de detecção de bordas dos fragmentos (como podemos ver ocorrendo no objeto D na figura 5.6), primeiro realçamos os objetos, dividindo a banda vermelha da imagem pela banda azul, pois como os cacos são de tonalidade marrom, o componente vermelho deve ser mais claro que o azul e, efetuando tal divisão, realçaríamos os objetos (figura 5.11).

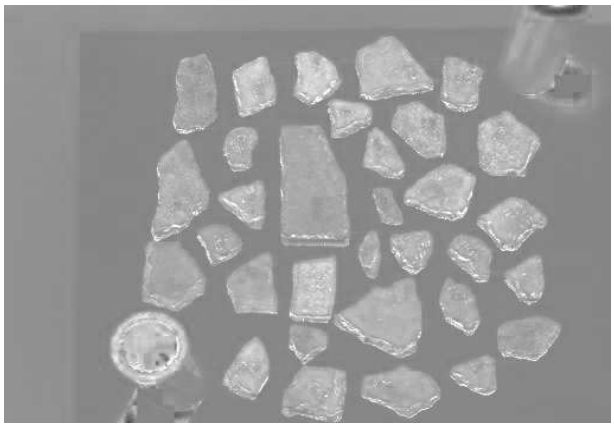
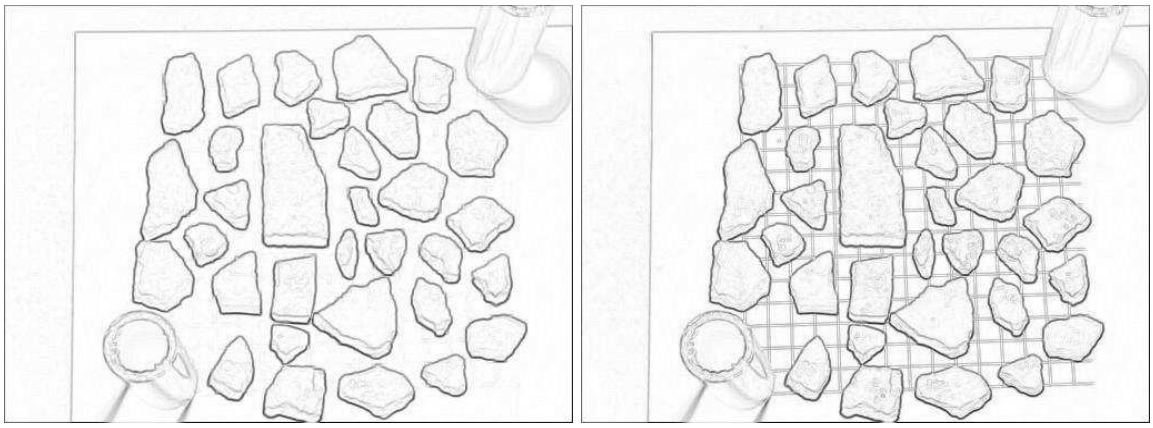


Figura 5.11: Imagem resultante do processo de realce dos fragmentos

5.3.2.2 Cálculo do gradiente

Como vimos no capítulo 4, a função de custos da IFT depende da magnitude de um gradiente, portanto, o modificamos o cálculo deste gradiente para o de Sobel [35], a fim de obtermos uma intensidade de contorno melhor (diferentemente do processo original), por se tratar de um bom gradiente [1, 12, 28, 65]. Portanto, não só calculamos o gradiente de Sobel, mas principalmente o seu complemento, ou seja, o valor de um pixel $p, I(p)$ é dado por $255 - I(p)$, que será utilizado como custo da aresta, que explicaremos melhor nas seções seguintes. A figura 5.12 ilustra este processo de aplicação do gradiente de Sobel.

Podemos observar que calculamos o gradiente de Sobel utilizando a imagem original e a imagem realçada, onde não aparecem as linhas de grade. Isto foi feito para se verificar se a remoção destas linhas pode afetar a eliminação das arestas espúrias.



(a) Gradiente 1, obtido da imagem realçada (b) Gradiente 2, obtido da imagem original

Figura 5.12: Imagens resultantes da aplicação do gradiente de Sobel.

5.3.2.3 Segmentação por limiarização

Para separar os objetos do restante da cena, usamos a imagem realçada 5.11 e sobre tal imagem, aplicamos uma limiarização ou Threshold [35] onde é escolhido o valor médio do histograma para separar os objetos. O objetivo desta etapa é obter uma imagem de orientação de borda, ou seja, uma imagem binária onde os fragmentos ou objetos de interesse estivessem evidenciados quanto a sua forma e suas fronteiras, como visto na figura 5.13. Desta forma, consideramos em nosso processo que o objeto é mais claro em seu interior do que fora, estabelecendo então a nossa orientação para a borda. No resultado mostrado na figura 5.13, podemos ver os fragmentos em branco e o restante em preto, posteriormente, essa imagem binária será importante também na resolução das arestas espúrias. Nosso objetivo, contudo, foi incorporar a orientação não pelo vetor gradiente de Sobel, mas por uma segmentação incompleta gerada por limiarização.

Figura 5.13: Imagem resultante do processo de limiarização

A segmentação por limiarização utilizada é aproximada, pois aparecem buracos, reentrâncias e fiapos. Em alguns objetos, há perda de informação, ou seja, algumas porções de alguns objetos (tais como cantos, arestas, dentre outras) não serão identificados. Isto causará diferença entre o contorno obtido e o contorno real do objeto. A fim de tentarmos resolver tais imperfeições, aplicamos uma função implementada por IFT chamada CloseHoles, cujo objetivo é identificar esses buracos e preenchê-los, resultando numa imagem sem buracos, embora não consiga recuperar a informação perdida. A figura 5.14 mostra a imagem resultante dessa aplicação, que a utilizaremos como imagem de orientação do contraste para a borda.

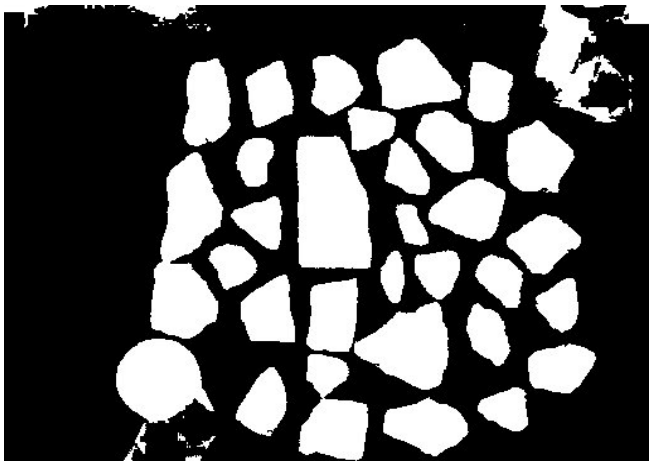


Figura 5.14: Imagem resultante da aplicação da IFT *CloseHoles*

Após o pré-processamento, modificamos a função de custos f_{ctrack}, como mostraremos a seguir.

5.3.3 Modificação na IFT para detecção de borda

Modificamos a função de custos para detecção de contornos usando a IFT, chamada F_{ctrack} [2] para a função $f_{track-modif}$. Esta função possui a seguinte forma:

$$f_{track-modif}(< t >) = 0;$$

$$f_{track-modif}(\pi \cdot < s,t >) = f_{track-modif}(\pi) + w(s,t)$$

onde:

$$w(s,t) = \begin{cases} G(t), \text{ se a orientação for satisfeita,} \\ 255, \text{ no caso contrário} \end{cases}$$

sendo $G(t)$ a magnitude do gradiente de Sobel em t e $w(s,t)$ é o peso da aresta (s,t).

Desta forma, alteramos o gradiente utilizado originalmente pelo gradiente de Sobel e utilizamos o seu complemento (seção 5.3.2.2) como custo de aresta. Seja (s,t) uma aresta e $G(t)$ o complemento do gradiente de Sobel no pixel t, então o peso $w(s,t)$ da aresta (s,t) será $G(t)$ se a orientação for satisfeita, e será 255 caso contrário. A função de custo é a soma dos pesos $w(s,t)$ das arestas (s,t) ao longo do caminho. Lembrando que a orientação a ser satisfeita é a orientação de contraste que estabelecemos anteriormente.

5.4 Metodologia dos testes

Nos testes realizados, utilizamos uma média de 4 pixels sementes, considerando a forma dos objetos de interesse. Como os fragmentos formam certos paralelogramos e vimos que necessitamos de um pixel semente para cada ponto de saliência, os pontos, portanto, estariam em cada vértice dos objetos o que nos dá o número suficiente citado acima.

Como na seção 5.3.2.3 apresentamos a imagem de orientação como sendo a imagem binarizada, entretanto uma alternativa seria utilizar a imagem de luminância. É denominado imagem de luminância a imagem que corresponde ao componente de luminância Y (informação de cinza) da imagem em cores, sendo uma das partes para a formação de cores da imagem [66]. Ou seja, utilizamos, portanto, a própria imagem de entrada como orientação, porém em tons de cinza.

Apesar de termos 2 possibilidades de utilização do gradiente de Sobel, observamos em testes preliminares que os resultados finais não eram afetados por tal escolha. Por conta disto, usamos em nossos testes o gradiente obtido pela imagem de realce (figura 5.12(a)).

Basicamente, foram realizados dois tipos de testes, onde são utilizados as diferentes imagens de orientação para a borda, mantendo-se a mesma função de custos modificada $f_{track-modif}$ e o gradiente calculado no pré-processamento:

1. Utilização da imagem binária (veja seção 5.3.2.3) como orientação para borda, onde assumimos que os objetos possuem um brilho maior dentro do que fora deles.

2. Utilização da imagem de luminância, onde assumimos que os objetos possuem um brilho menor dentro do que fora deles.

Ao optarmos pelo segundo tipo de teste, procuramos excluir os erros que a perda de informação na imagem de orientação por limiarização causa, quando se traça o contorno de alguns objetos. O objetivo seria avaliar esta segunda opção e observarmos a execução

do método somente através da alteração da função de custos e utilização de um gradiente forte. O resultados obtidos serão apresentados no próximo capítulo.

Capítulo 6

Experimentos e Resultados

Neste capítulo, apresentaremos os resultados obtidos para todas as imagens de teste, a fim de se resolver os problemas e soluções propostas no capítulo anterior para a aplicação de interesse. Juntamente com os resultados, faremos uma análise dos mesmos.

6.1 Resultados utilizando imagem binária

A seguir, temos os resultados dos testes utilizando como orientação para a borda a imagem binária obtida por limiarização (conforme descrito na seção 5.3.2.3) para as 5 imagens (figura 5.1 a figura 5.5) do capítulo anterior:

Figura 6.1: Imagem contendo a identificação dos objetos da figura 5.1 para a 1ª abordagem dos testes.

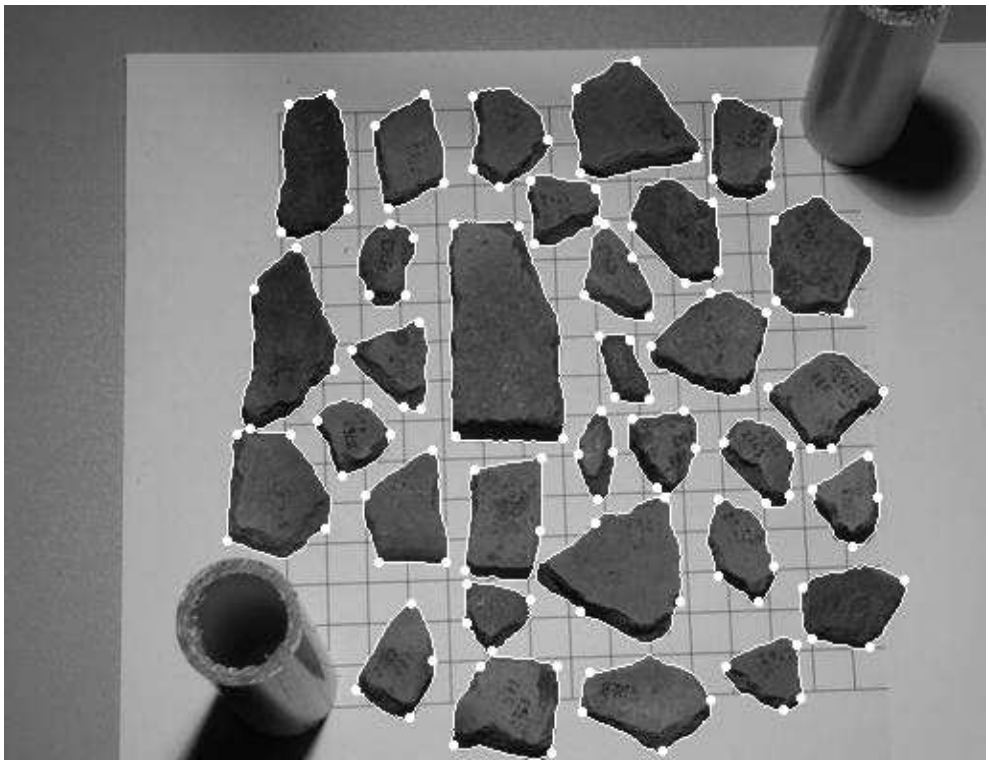


Figura 6.2: Imagem contendo a identificação dos objetos da figura 5.2 para a 1ª abordagem dos testes.

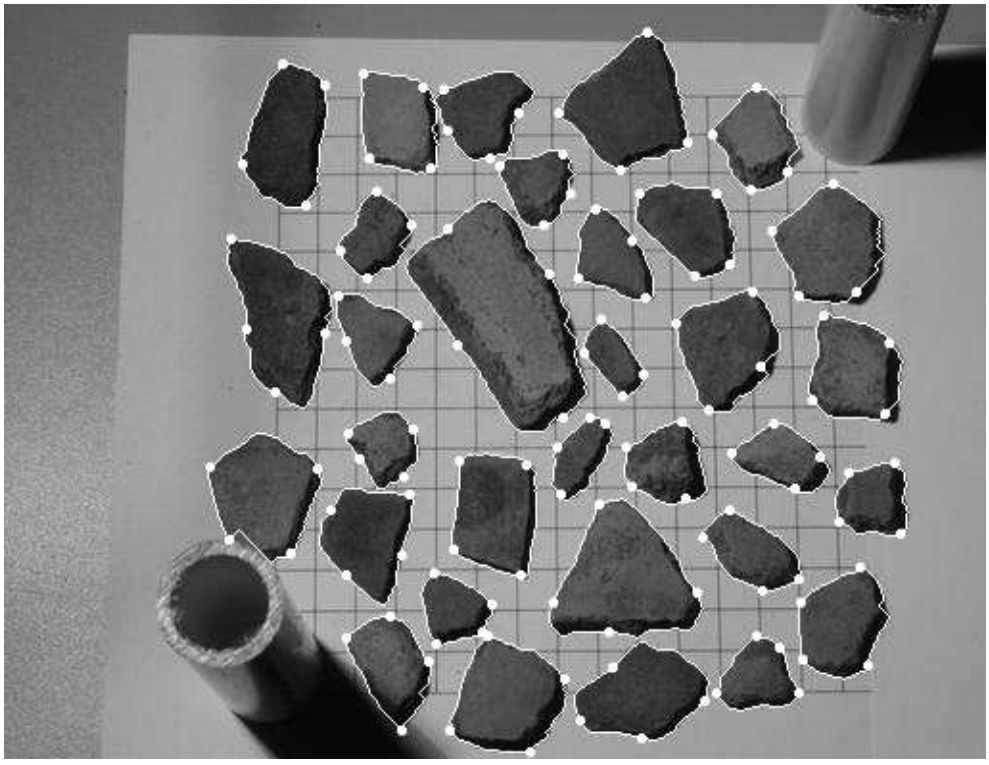


Figura 6.3: Imagem contendo a identificação dos objetos da figura 5.3 para a 1ª abordagem dos testes.

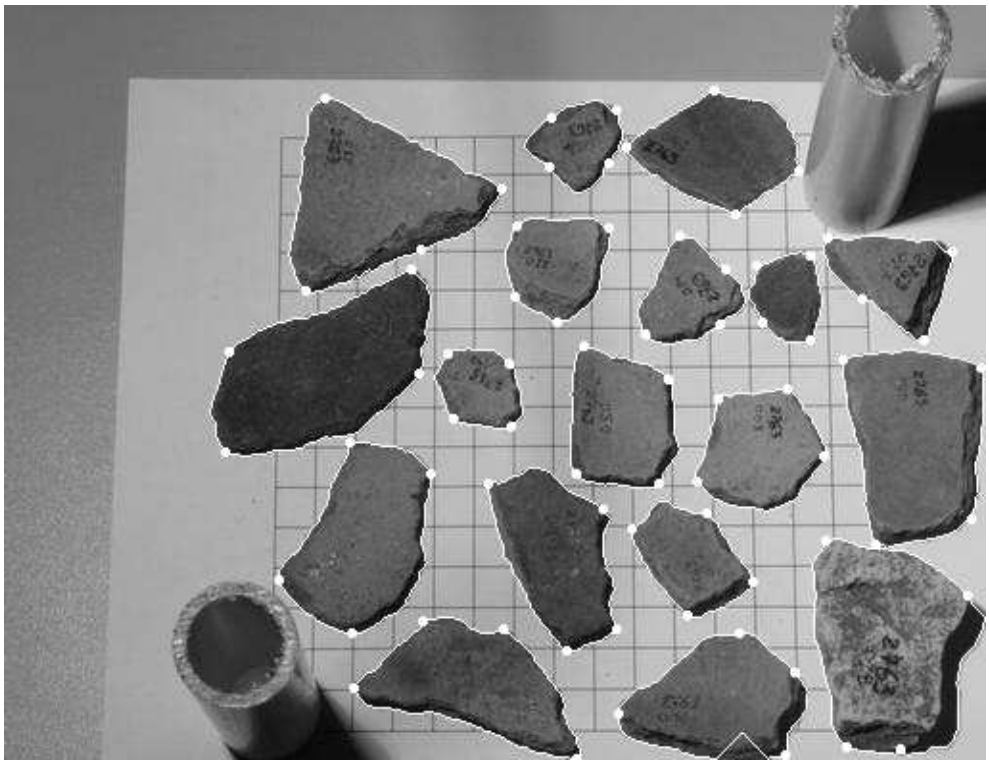


Figura 6.4: Imagem contendo a identificação dos objetos da figura 5.4 para a 1ª abordagem dos testes.

Figura 6.5: Imagem contendo a identificação dos objetos da figura 5.5 para a 1ª abordagem dos testes.

Para o conjunto de imagens acima, temos a seguinte tabela:

Figura	**Nº de objetos**	**Acertos**	**Falhas**
01	34	18	16
02	32	32	0
03	32	19	13
04	18	16	2
05	19	14	5

Tabela 6.1: Resultados obtidos utilizando orientação pela imagem binária

Contudo, observamos que em alguns objetos, o traçado do contorno se mostra inconsistente, devido a influência da orientação pela imagem binária. Vimos no capítulo 5 que a segmentação por limiarização é suscetível a falhas, ocorrendo perda de informação em alguns objetos. Essa característica, portanto, influenciou nos nossos testes, produzindo bordas irregulares em alguns objetos. Este fato ficou mais evidente nas figuras 01 e 03, onde a imagem binária ficou comprometida talvez pela influência do maior número de sombras ou iluminação mais deficiente. Pode-se observar também que nestas imagens, além do maior número de objetos, estes estão mais próximos e, portanto, isto pode ser outro fator de influência, quando da separação dos objetos no processo de limiarização.

Em objetos que estão muito colados um ao outro (como na figura 6.1), a busca por sua borda fica comprometida, já que a parte em contato não fica evidente o suficiente para o seu correto reconhecimento. Do mesmo modo, objetos em que parte de sua forma é sobreposta por outros elementos (na figura 6.3 pelo gabarito) ou que ocorreu uma perda de sua forma (figura 6.4), o traçado de sua borda também é comprometido pelo mesmo motivo anterior.

6.2 Resultados utilizando imagem de luminância

Nesta seção, apresentaremos os resultados obtidos no segundo método de teste, onde foi utilizado como orientação para a borda a imagem de luminância, conforme apresentado na seção 5.4, para o mesmo conjunto de imagens 5.1, 5.2, 5.3, 5.4, 5.5:

Figura 6.6: Imagem contendo a identificação dos objetos da figura 5.1 para a 2ª abordagem dos testes.

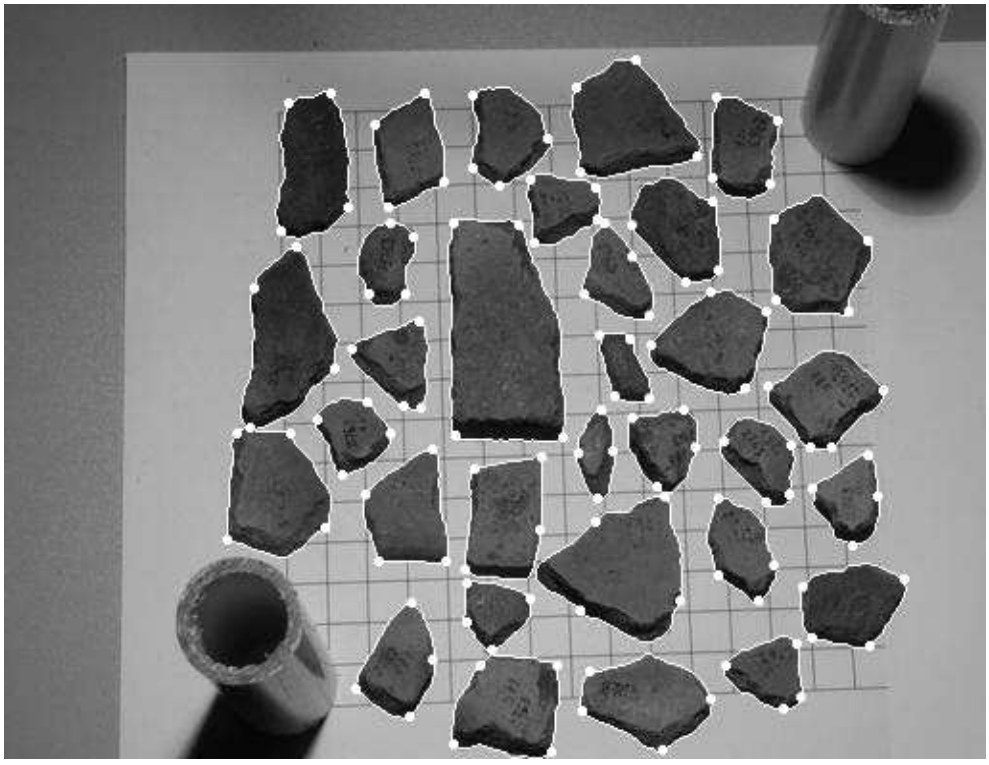


Figura 6.7: Imagem contendo a identificação dos objetos da figura 5.2 para a 2ª abordagem dos testes.

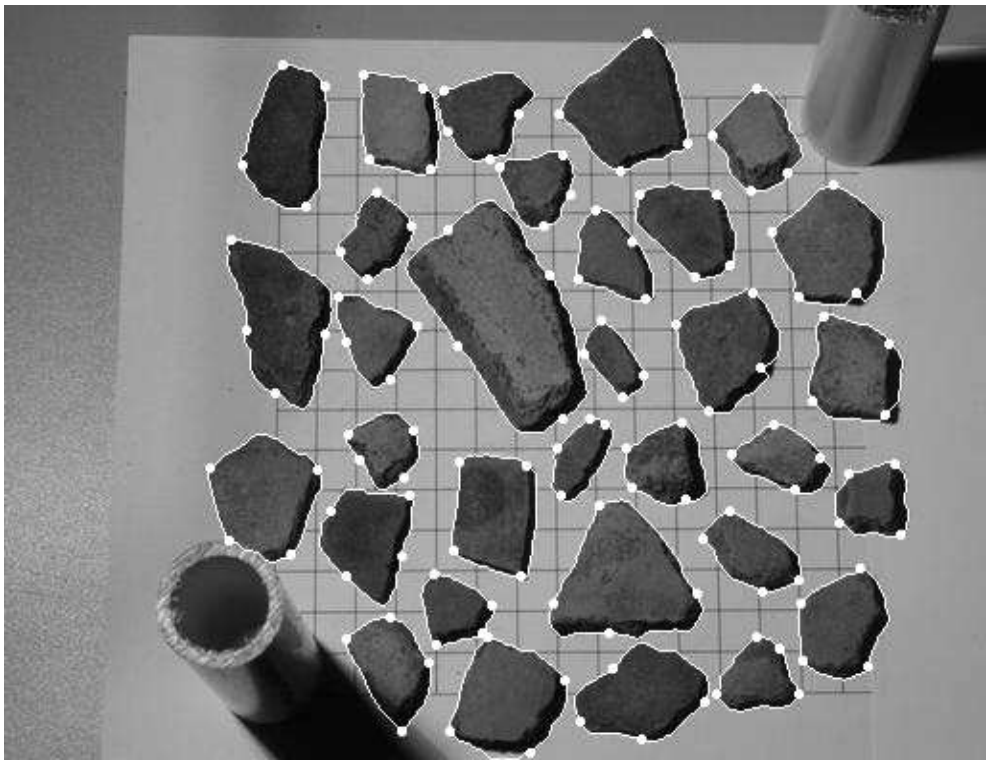


Figura 6.8: Imagem contendo a identificação dos objetos da figura 5.3 para a 2ª abordagem dos testes.

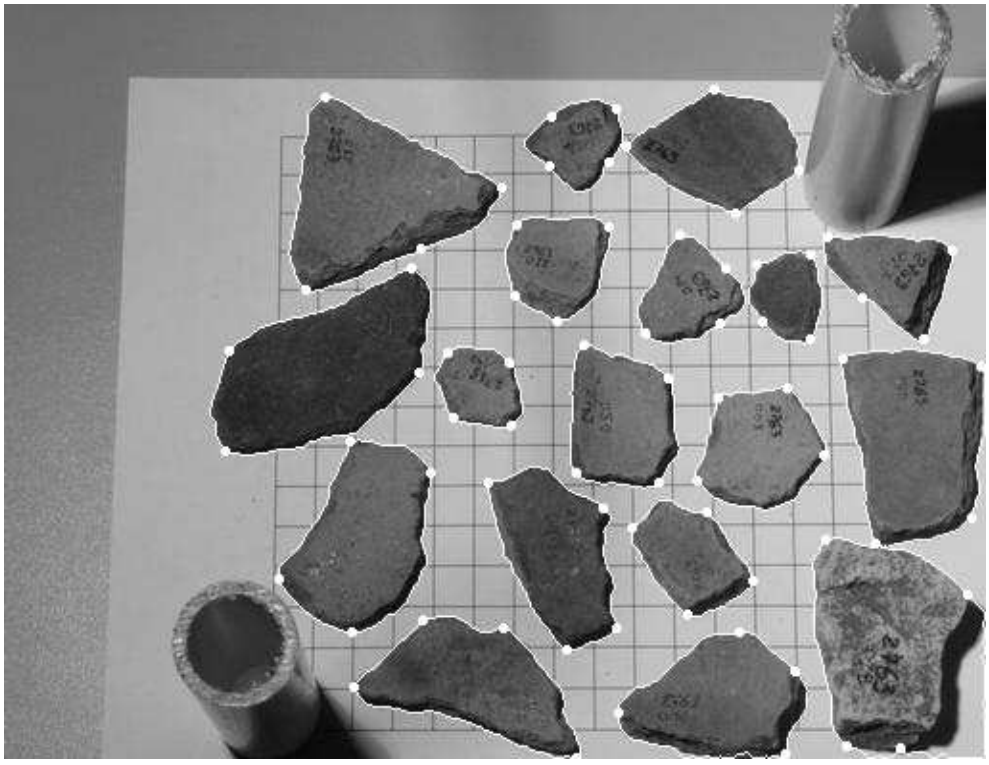


Figura 6.9: Imagem contendo a identificação dos objetos da figura 5.4 para a 2ª abordagem dos testes.

Figura 6.10: Imagem contendo a identificação dos objetos da figura 5.5 para a 2ª abordagem dos testes.

Para o conjunto de imagens acima, temos a seguinte tabela:

Figura	**Nº de objetos**	**Acertos**	**Falhas**
01	34	32	2
02	32	32	0
03	32	29	3
04	18	16	2
05	19	19	0

Tabela 6.2: Resultados obtidos utilizando orientação pela imagem de luminância

Nas figuras 6.6, 6.7, 6.8, 6.9 e 6.10, podemos observar a melhoria do traçado obtido, onde percebemos que o traço parece "colado" à borda, além se ter um aspecto mais suave, o que seria o desejado e caracterizando também sua eficiência.

Os insucessos nesta abordagem ocorreram mais pela sobreposição e perda de informação dos objetos (como foi explicado na seção anterior) do que pela má detecção de borda. Neste aspecto, esta abordagem se mostrou mais eficaz, apresentando melhor exatidão.

Nesta segunda abordagem, além de uma simplificação do processo, ao se excluir a necessidade da utilização da segmentação por limiarização externa ao método.

6.3 Conclusões dos testes

Um alto valor do gradiente na borda do objeto é imprescindível como visto na seção 5.3.3, pois a modificação realizada na função de custos depende do valor do gradiente em algum ponto da borda. Com isso, as arestas espúrias são solucionadas, pois o traçado respeita com exatidão os pixels na borda dos objetos, não ocorrendo somente nos casos onde os objetos estiverem colados, isto é, encostados uns aos outros, o que por condições óbvias, não se permite claramente a identificação dos limites dos objetos em questão.

A utilização da imagem de luminância é suficiente para a identificação dos objetos, não justificando o uso da limiarizaçõa para o conjunto de imagens testado.

Considerando, portanto, o pré-processamento (seção 5.3.2) e a mudança da função de custos da IFT (seção 5.3.3) para o traçado do contorno, além de ter solucionado as arestas espúrias, observamos uma relevante melhora da qualidade do contorno de bordas, no que tange a exatidão de detecção das mesmas nos objetos.

Capítulo 7

Considerações Finais e Trabalhos Futuros

Neste trabalho estudamos o método da IFT com ênfase da segmentação por perseguição de contornos (bordas).

Na segmentação para perseguição de contornos, vimos que a IFT obtém resultados eficazes e de qualidade, podendo, portanto, ser aplicado a diversas áreas e pesquisas que necessitam desse tipo de resultado, sem perdas de informações importantes ou mesmo comprometendo o resultado final das imagens.

Porém, observamos que, para objetos muito próximos, ocorria o problema do aparecimento de arestas indesejadas. Apresentamos, então, uma solução que resolve este problema, sendo modificada a função de custos original da IFT para o propósito em questão. Nos testes realizados, vimos que a utilização da orientação do traçado pela magnitude do gradiente foi suficiente, produzindo resultados eficazes e descartando a necessidade da utilização da imagem binária como orientação.

Mais testes necessitam ser feitos no intuito de se verificar a real necessidade e importância do pré-processamento realizado, sobretudo na obtenção da imagem de orientação por limiarização (imagem binária). A variação das características das amostras poderia ser aplicado para se observar se a simples modificação da função de custos e adoção de um bom gradiente seriam suficientes para um processo eficiente e seguro, ou se a imagem de orientação obtida pela limiarização faz-se necessária em outros determinados processos.

É importante na IFT a escolha de pixels sementes em todas as aplicações de segmentação que ela executa, principalmente na detecção de bordas, e observamos que esta tarefa é feita manualmente na imagem de interesse, fazendo uso de aplicativos de manipulação de imagens, o que torna o processo que antecede a aplicação da IFT trabalhoso, pouco usual e lento. Sendo assim, haveria a necessidade de se implementar um algoritmo de busca

de pixels sementes na imagem, e pensando na detecção de contornos, pixels sementes na borda dos objetos, sobretudo em pontos de alta curvatura (seção 5.2). Portanto, basicamente, o algoritmo deveria realizar a investigação e busca desses pontos, transformando-os em sementes da IFT na detecção do contorno desejado.

Além da escolha dos pixels sementes, a orientação dos mesmos é crucial para uma segmentação correta e talvez, o uso de uma heurística simples seja possível obter resultados satisfatórios para algumas imagens.

Nos nossos estudos, foram realizados também testes preliminares utilizando crescimento de regiões para obtenção do contorno dos objetos. Assim, utilizamos a IFT para segmentação baseada na transformada Watershed [2]. Apesar de nosso foco estar em segmentação por contorno de objeto, procuramos num primeiro momento comparar os resultados obtidos em nosso método em detrimento àqueles fornecidos pela Watershed, observando não só a qualidade dos resultados alcançados, como as diferenças entre as implementações dos algoritmos. Alguns contornos traçados pela IFT-Watershed em alguns objetos não foram traçados com eficiência, como mostrado pela figura 7.1.

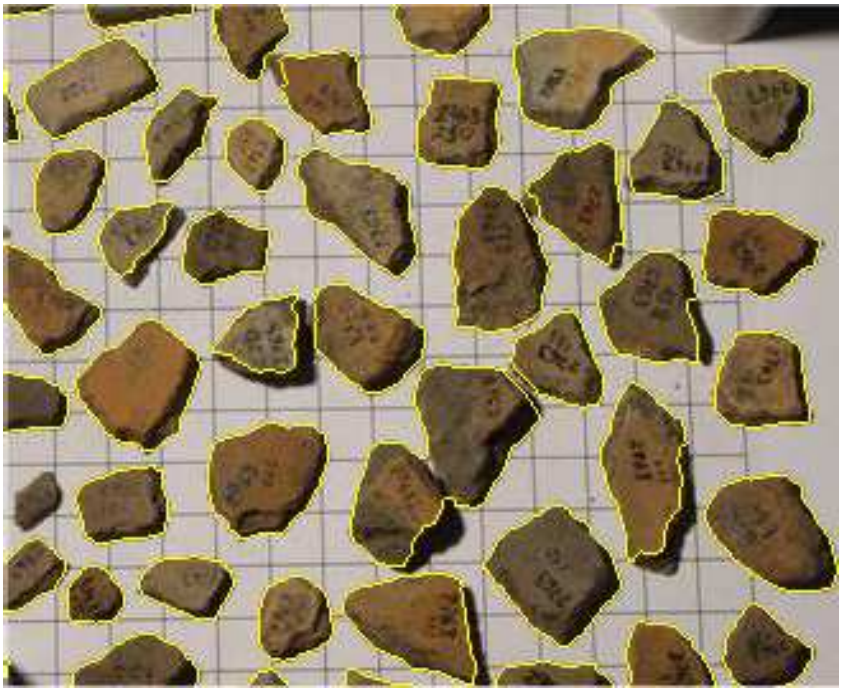


Figura 7.1: Imagem resultante da aplicação da IFT *CloseHoles*

Estas falhas ocorridas foram sensíveis a basicamente dois fatores: a escolha do gradiente e o efeito das sombras ocorridas pela posição da luz da obtenção das imagens.

Em termos de implementação, notamos que devemos ter uma atenção especial na busca de marcadores internos aos objetos. Conseguimos implementar essa busca de maneira global e automática, ou seja, identificamos os objetos e automaticamente seus marcadores internos. Aplicamos a função Watershed para todos os objetos da imagem, detectando seus contornos. Portanto, uma das vantagens da Watershed é a escolha automática das sementes. Em contrapartida, a vantagem de nosso método é sua invariância quanto a escolha dos gradientes e efeito das sombras.

Referências

[1] GONZALES, R. C.; WOODS, R. E. *Digital Image Processing*. Ed.: Addison-Wesley Publishing Company, 1993.

[2] FALCÃO, A. X.; STOLFI, J.; LOTUFO, R. A. The image foresting transform: Theory, algorithms, and applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, IEEE Press, v. 26, n. 1, p. 19–29, Jan 2004.

[3] COCQUEREZ, J. P.; PHILIPP, S. *Analyse d'Images: Filtrage et Segmentation*. Paris: Masson S. A., 1995.

[4] CANNY, J. F. A computational approach to edge detection. *Readings in Computer Vision: Issues, Problems, Principles and Paradigms. M. A Fischer and Brian*, Morgan Kaufmann, v. 1, n. 1, p. 184–203, 1986.

[5] MEYER, F.; BEUCHER, S. The morphological approach to segmentation: The watershed transform. *Mathematical Morphology in Image Processing*, Dogherty, v. 1, n. 1, p. 433–481, set. 1992.

[6] COHEN, L. D.; COHEN, I. Finite- element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 15, p. 11, 1993.

[7] BRINKLEY, J. A flexible, generic model for anatomic shape: application to interactive two-dimensional medical image segmentation and mathing. *Computers on Biomedical Research*, v. 26, p. 121–142, 1993.

[8] CHALANA, W. C. V.; KIM., Y. Integrating region growing and edge detection using regularization. *In Proceddings of the SPIE Conference on Medical Imaging*, v. 15, p. 11, 1995.

[9] LAI, K. F. *Deformable Contours: Modeling, Extraction, Detection and Classification*. Tese (Doutorado) — University of Wisconsin-Madison, Madison, 1994.

[10] OSHER, S.; PARAGIOS, N. *Geometric Level Set Methods in Imaging,Vision,and Graphics*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003. ISBN 0387954880.

[11] FALCÃO, A. X.; CUNHA, B. S. da; LOTUFO, R. A. Design of connected operators using the image foresting transform. In: *Proc. of SPIE on Medical Imaging*. [S.l.: s.n.], 2001. v. 4322, p. 468–479.

[12] FALCÃO, A. X.; UDUPA, J. K. A 3D generalization of user-steered live wire segmentation. *Medical Image Analysis*, Elsevier, v. 4, n. 4, p. 389–402, Dec 2000.

[13] FALCÃO, A. X.; BERGO, F. P. G. Interactive volume segmentation with differential image foresting transforms. *IEEE Trans. on Medical Imaging*, IEEE Press, v. 23, n. 9, p. 1100–1108, Sep 2004.

[14] FALCÃO, A. X.; COSTA, L. F.; CUNHA, B. S. da. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, Elsevier, v. 35, n. 7, p. 1571–1582, Apr 2002.

[15] NEVES, S. C. M.; PELAES, E. G. Estudo e implementação de técnicas de segmentação de imagens. *Revista Virtual da Iniciação Científica*, Universidade Federal do Pará, v. 02, 2001.

[16] RIBEIRO, C.; HEMERLY, E. Segmentação de imagens: Algoritmos e aplicações. *Anais do XII Seminário de Ciências Exatas e Engenharias da UNESP*, v. 01, p. 70, 1992. Guaratinguetá-SP.

[17] PARKER, J. R. *Algorithms for Image Processing and Computer Vision*. Ed.: John Wiley and Sons, 1997.

[18] BURNHAM, J. et al. A comparison of the roberts, sobel, robinson, canny, and hough image detection algorithms. *IEEE Southeastcon 98*, v. 01, 1998.

[19] HARALICK, R. M.; SHAPIRO, L. G. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, v. 29, n. 1, p. 100–132, jan. 1985. ISSN 0734-189X. HARALICK85.

[20] ZHANG, Y. J. A survey on evaluation methods for image segmentation. *Pattern Recognition Letters*, v. 29, p. 1335–1346, 1996.

[21] MOREL, J. M.; SOLIMINI, S. *Variational methods in image segmentation*. Cambridge, MA, USA: Birkhauser Boston Inc., 1995. ISBN 0-8176-3720-6.

[22] MANCAS, M.; GOSSELIN, B. Segmentation using a region growing thresholding. *Image Processing: Algorithms and Systems IV*, SPIE The International Society for Optical Engineering and IS&T The Society for Imaging Science and Technology, v. 5672, n. 5672, p. 388–398, 2005.

[23] NASCIMENTO, P. *Avaliação de técnicas de segmentação e classificação por regiões em imagens Landsat-TM visando o mapeamento de unidades de paisagem na Amazônia*. Dissertação (Mestrado) — INPE-6391- TDI/607, São José dos Campos-SP, Brasil, 1997.

[24] ZHANG, Y. J. Evaluation and comparison of different segmentation algorithms. *Pattern Recognition Letters*, v. 18, p. 963–974, 1997.

[25] DEJHAN, K. et al. A sobel edge detector digital filter structure and its distributed arithmetic implementation. *ACRS - Asian Conference on Remote Sensing*, v. 01, 1995.

[26] CHEN, F.; DAVID, S. Multiscale image representation and edge detection. In: *ACCV '98: Proceedings of the Third Asian Conference on Computer Vision-Volume II*. London, UK: Springer-Verlag, 1997. p. 49–56. ISBN 3-540-63931-4.

[27] ZIOU, D.; TABBONE, S. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, v. 8, n. 4, p. 537–559, 1998.

[28] ALBUQUERQUE, M. P. *Analyse par Traitement D'Images.* Tese (Doutorado) — Laboratoire de Magnetism Louis Néel, Grenoble, 1995.

[29] UDUPA, J. K.; SAMARASEKERA, S. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Graph. Models Image Process.*, Academic Press, Inc., Orlando, FL, USA, v. 58, n. 3, p. 246–261, 1996. ISSN 1077-3169.

[30] MARQUES, O.; VIEIRA, H. *Processamento Digital de Imagens.* Ed.: Brasport Livros e Multimídia Ltda, 1992.

[31] SONKA, M.; HLAVAC, V.; BOYLE, R. *Image Processing: Analysis and Machine Vision.* [S.l.]: O'Reilly, 1999. SON m 99:1 1.Ex.

[32] PACIORNIK, S. *Introdução ao Processamento Digital de Imagens.* Julho 2004. Disponível em: <http://www.dcmm.puc-rio.br/Cursos/IPDI/tsld116.htm>.

[33] CAMAPUM, J. F.; FISHER, M. H. Segmentation using spatial-feature clustering from image sequences. In: *ICIP (3).* [S.l.: s.n.], 1998. p. 799–803.

[34] SAVAKIS, A. E. Adaptive document image thresholding using foreground and background clustering. In: *ICIP (3).* [S.l.: s.n.], 1998. p. 785–789.

[35] WATT, A.; POLICARPO, F. *The Computer Image.* Ed.: Addison-Wesley, 1998. ISBN 0201422980.

[36] LEYMARIE, F.; LEVINE, M. Tracking deformable objects in the plane using an active contour model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, v. 15, p. 617–633, 1993.

[37] LOBREGT, S.; VIERGEVER, M. A. A discrete dynamic contour model. *IEEE Transactions on Medical Imaging*, v. 14, p. 1, 1995.

[38] MACKIEWICH, B. *Active contour models (snakes).* Julho 1995. Disponível em: <http://www.cs.sfu.ca/ stella/papers/blairthesis/main/node29.htm>.

[39] NUENSCHWAANDER, P. F. W.; KUBLER, O. Initializing snakes. *IEEE Conference on Computer Vision and Pattern Recognition*, v. 1, p. 658–663, 1994.

[40] KAAS, A. W. M.; TERZOPOULOS, D. Snakes: Active contour models. *International Journal of Computer Vision*, v. 1, p. 321–331, 1987.

[41] COHEN, L.; COHEN, I. Using a finite element method for active contour models and 3d reconstruction from cross sections. *Proc. Third International Conference on Computer Vision*, v. 1, p. 587–591, 1990.

[42] COHEN, L.; COHEN, I. Using deformable surfaces to segment 3d images and infer differential structures. *CVGIP: Image Understanding*, v. 56, p. 242–263, 1992.

[43] RODRIGUES, E. A. *Segmentação de Imagens Tomográficas usando Contornos Deformáveis com Segmentos de Custo Mínimo.* Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas-SP, Brasil, 2002. Instituto de Computação.

[44] MCINERNEY, T.; TERZOPOULOS, D. Deformable models in medical image analysis: a survey. *Medical Image Analysis*, v. 1, p. 91–108, 1996.

[45] BOISSONNAT, J. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing*, v. 1, p. 1–29, 1988.

[46] BERGER, M.; MOHR, R. Towards autonomy in active contour models. *Proceedings of the Tenth International Conference on Pattern Recognition*, v. 1, p. 847–851, 1990.

[47] LEYMARIE, F.; LEVINE, M. Simulating the grassfire transform using an active contour model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, v. 14, p. 56–75, 1992.

[48] AMINI, T. W. A. A.; JAIN, R. Using dynamic programming for solving variatonal problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 12, p. 855–867, 1990.

[49] ZUCKER, A. D. S.; IVERSON, L. Two stages of curve suggest two styles of visual computation. *Neural Computation*, v. 1, p. 68–81, 1989.

[50] COOPER H. ELLIOT, F. C. L. R. D.; SYMOSEK, P. Stochastic boundary estimation and object recognition. *Computer Graph. and Image Proc.*, v. 12, p. 326–356, 1980.

[51] DIGABEL, H.; LANTUÉJOUL, C. Iterative algorithms. In: CHERMANT, J.-L. (Ed.). *Quantitative analysis of microstructures in materials sciences, biology and medicine*. Stuttgart: Dr. Riederer-Verlag GmbH, 1978. p. 85–99.

[52] BEUCHER, S.; LANTUÉJOUL, C. Use of watersheds in contour detection. In: *International Workshop on Image Processing*. Rennes: CCETT/IRISA, 1979. p. 2.1–2.12. Disponível em: <http://cmm.ensmp.fr/~beucher/publi/watershed.pdf>.

[53] DEO, N.; PANG, C. *Shortest path algorithms: Taxonomy and annotation.* [S.l.]: Networks, 1984. 275-323.

[54] HARARY, F. *Graph Theory*. 10. ed. Reading: Perseus Books, 1998. ISBN 0-201-41033-8.

[55] TRUDEAU, R. J. *Introduction to Graph Theory*. Ed: Dover Publications, 1993. ISBN 0-486-67870-0.

[56] AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. *Network Flows: Theory, Algorithms, and Applications*. New Jersey: Prentice Hall, 1993. AHU r 93:1 P-Ex.

[57] CORMEN, T.; LEISERSON, C.; RIVEST, R. *Introduction to Algorithms*. [S.l.: s.n.], 1989. xvii + 1028 p. ISBN 0-262-03141-8.

[58] DIAL, R. B. Algorithm 360: Shortest-path forest with topological ordering [h]. *Commun. ACM*, ACM Press, New York, NY, USA, v. 12, n. 11, p. 632–633, 1969. ISSN 0001-0782.

[59] FALCÃO, A. X.; UDUPA, J. K.; MIYAZAWA, F. K. An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly. *IEEE Trans. on Medical Imaging*, v. 19, n. 1, p. 55–62, Jan 2000.

[60] RAGNEMALM, I. Neighborhoods for distance transformations using ordered propagation. *CVGIP: Image Underst.*, Academic Press, Inc., Orlando, FL, USA, v. 56, n. 3, p. 399–409, 1992. ISSN 1049-9660.

[61] DANIELSSON, P.-E. Euclidean distance mapping. *Computer Graphics and Image Processing, volume = 14, pages = 227–248, OPTnote = CGIP, Academic Press.*, 1980.

[62] CUISENAIRE, O.; MACQ, B. Fast euclidean distance transformation by propagation using multiple neighborhoods. *Comput. Vis. Image Underst.*, Elsevier Science Inc., New York, NY, USA, v. 76, n. 2, p. 163–172, 1999. ISSN 1077-3142.

[63] RAAB. *Reconstrução de Artefatos Arqueológicos Brasileiros.* Instituto de Arqueologia Brasileira - IAB, Março 2005. Disponível em: <http://www.ic.uff.br/~raab/raab-index.html>.

[64] PARKS, D.; GRAVEL, J. P. Corner detectors. Http://www.cim.mcgill.ca/~dparks/index.htm. Julho 2005.

[65] MARTA, D. S. Algoritmos para detecção de bordas. UFSC, Brasil. 1998.

[66] FACON, J. *Princípios Básicos da Visão por Computador e do Processamento de Imagens.* Paraná: Pontifícia Universidade Católica do Paraná, 2002.