Segmentação Interativa de Volumes Baseada em Regiões

Felipe Paulo Guazzi Bergo

Dissertação de Mestrado

Segmentação Interativa de Volumes Baseada em Regiões

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Felipe Paulo Guazzi Bergo e aprovada pela Banca Examinadora.

Campinas, 13 de Fevereiro de 2004.

Prof. Dr. Alexandre Xavier Falcão
Instituto de Computação – UNICAMP
(Orientador)

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

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bergo, Felipe Paulo Guazzi

B454s Segmentação interativa de volumes baseada em regiões / Felipe Paulo Guazzi Bergo -- Campinas, [S.P. :s.n.], 2004.

Orientador : Alexandre Xavier Falcão

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

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

Instituto de Computação
Universidade Estadual de Campinas

Segmentação Interativa de Volumes Baseada em Regiões

Felipe Paulo Guazzi Bergo

Fevereiro de 2004

Banca Examinadora:

- Prof. Dr. Alexandre Xavier Falcão
  Instituto de Computação – UNICAMP (Orientador)
- Prof. Dr. Roberto de Alencar Lotufo
  Faculdade de Engenharia Elétrica e de Computação – UNICAMP
- Prof. Dr. Marcos Cordeiro d'Ornellas
  Departamento de Eletrônica e Ciência da Computação – Universidade Federal de Santa Maria
- Prof. Dr. Siome Klein Goldenstein
  Instituto de Computação – UNICAMP
- Prof. Dr. Jorge Stolfi (Suplente)
  Instituto de Computação – UNICAMP
- Prof. Dr. Fernando Cendes (Suplente)
  Faculdade de Ciências Médicas – UNICAMP

.

Prefácio

Segmentação de imagens é um problema subjetivo que, em geral, requer intervenção do usuário. O uso arraigado de imagens tri-dimensionais do corpo humano, provenientes de ressonância magnética (MR) ou tomografia computadorizada (CT), para diagnóstico, treinamento e planejamento de cirurgias criou uma demanda para métodos rápidos, eficientes e confiáveis de segmentação de imagens tri-dimensionais (volumes).

Neste trabalho estendemos a transformada imagem-floresta (IFT) para permitir a segmentação de volumes com correções interativas, desenvolvemos procedimentos de pré-processamento para permitir a segmentação de algumas estruturas de interesse no cérebro, a partir de imagens de MR, e apresentamos um método de "rendering" suficientemente rápido para prover visualização 3D durante a segmentação sem comprometer a interatividade. Como vantagem, alguns erros que poderiam ser cometidos pelo usuário durante a segmentação são evitados com a informação 3D da anatomia dos órgãos durante a segmentação.

O método proposto neste trabalho — a IFT diferencial (DIFT) — permite uma redução de 90% no tempo de processamento necessário para a segmentação e reduz o tempo de espera do usuário em cada correção interativa de 20 para 3 segundos.

Abstract

Image segmentation is a subjective problem, where user intervention is often required. The widespread use of tri-dimensional images of the human body, obtained with Magnetic Resonance (MR) or Computed Tomography (CT), for diagnostic, training and surgery planning generated a demand for fast, efficent and trustable methods for tri-dimensional image (volume) segmentation.

In this work we extend the image foresting transform (IFT) to allow volume segmentation with interactive corrections, devise pre-processing procedures for segmentation of some structures of the brain from MR images, and present a rendering method fast enough to provide 3D visualization during segmentation without compromising its interactivity. Interactive 3D visualization allows for the user to choose better segmentation markers over the anatomic structures, avoiding mistakes during the segmentation task.

The method introduced in this work — the differential IFT (DIFT) — leads to a 90% reduction of the processing time required for segmentation, and reduces the user's waiting time for each interactive correction from 20 to 3 seconds.

Agradecimentos

A todos que contribuiram diretamente no desenvolvimento deste trabalho: Alexandre Falcão (Orientador) e o corpo docente do Instituto de Computação da Unicamp, e a todos os pesquisadores do Laboratório de Neuro-Imagem do Hospital das Clínicas da Unicamp.

À minha mãe pelo total apoio dado não apenas durante este trabalho, mas ao longo de todo o caminho que me levou até aqui.

Àqueles que contribuiram para este trabalho nas mais improváveis e inesperadas formas: Daniel, Adriano e Dante; Roger, Syd, David, Richard e Nick; Ken, Hideaki e Megumi.

À CAPES pelo apoio financeiro.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A obtenção de imagens do corpo humano através de ressonância magnética (MR, do inglês *Magnetic Ressonance*) e tomografia computadorizada (CT, do inglês *Computerized Tomography*) tornou-se rotina em ambientes clínicos, com aplicação direta no diagnóstico de uma grande variedade de patologias, evitando a necessidade de intervenções invasivas [50]. Um exame de MR/CT consiste de uma seqüência de imagens de corte (fatias) ao longo de uma região do corpo do paciente, formando um volume de dados ou imagem tri-dimensional. Entretanto, essas imagens obtidas são em geral sub-utilizadas: o uso mais comum é a visualização isolada das fatias, ignorando a característica tri-dimensional dos dados.

Nas décadas de 1980 e 1990 uma grande variedade de métodos foi proposta para MR e CT. Entretanto, os métodos propostos eram lentos, não confiáveis e requeriam grande compreensão por parte do usuário, criando uma barreira para seu uso por médicos. Esses métodos têm evoluído bastante desde 1990, e hoje existem várias aplicações de processamento de imagens sendo usadas em ambientes clínicos. Entretanto, a interface homem-computador desses softwares continua dificultando seu uso por parte dos médicos devido à grande quantidade de operações e nomenclatura utilizada para as operações.

Um procedimento muito desejado em ambientes clínicos é a segmentação, que mapeia (classifica) cada elemento de volume de dados (voxel) em um objeto definido pelo usuário ou por uma aplicação. A segmentação permite diversos tipos de análise, tais como medição de volume, comparação entre pacientes e correlação em bancos de dados. A segmentação é um dos principais desafios em processamento de imagens [15]. No caso de imagens médicas, em particular, a segmentação freqüentemente requer intervenção do usuário [16, 22, 27, 26, 47], e a falta de um método eficiente para uma dada aplicação faz com que o tempo necessário para a segmentação manual seja muito elevado (em torno de 20 minutos para um volume pequeno, com 30 fatias), tornando o uso do processamento de imagens médicas inviável na rotina de clínicas e hospitais

Neste trabalho estendemos a transformada imagem-floresta (IFT, do inglês *Image Foresting Transform*) para realizar a segmentação interativa de volumes através da aplicação de operações diferenciais com visualização 3D simultânea.

Nosso principal objetivo é propor um método de segmentação de volumes guiado pelo usuário, que permite executar a segmentação em tempo razoável (Em torno 10 minutos por objeto segmentado, em volumes com 100–180 fatias), com qualidade aceitável por médicos especialistas, e que não requer compreensão do usuário sobre a teoria envolvida no método. Com a IFT diferencial (método proposto neste trabalho) conseguimos realizar segmentação interativa de volumes em PCs de baixo custo, com uma redução de 90% do tempo de processamento em relação à IFT não diferencial, e redução semelhante no tempo de resposta ao usuário (de 20 para 3 segundos).

1.1 Organização do Trabalho

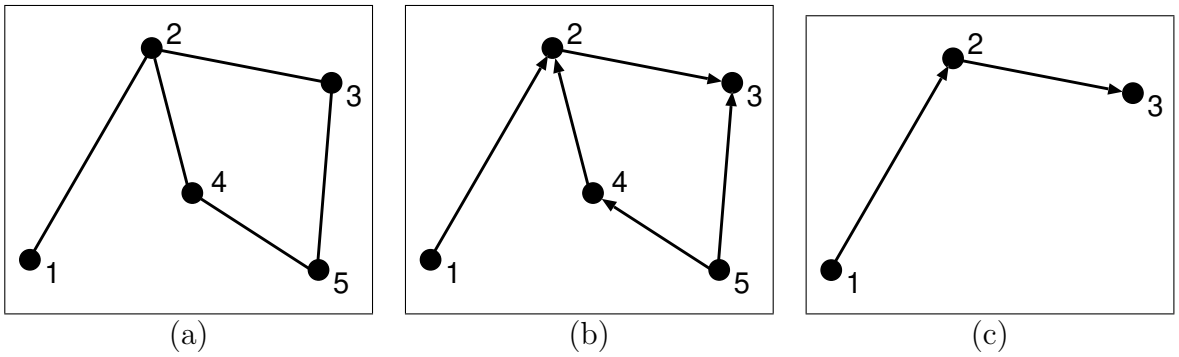
Neste capítulo apresentamos conceitos e notações referentes a teoria dos grafos, computação gráfica e processamento de imagens que serão usados ao longo dos capítulos seguintes. No capítulo 2 introduzimos a transformada imagem-floresta. No capítulo 3 estendemos a IFT para realizar atualizações diferenciais de forma eficiente. No capítulo 4 apresentamos alguns métodos de segmentação baseados na IFT diferencial. No capítulo 5 apresentamos os resultados obtidos com a aplicação da IFT diferencial na segmentação interativa de imagens médicas. O capítulo 6 discute a implementação de um método de visualização adequado a aplicações de segmentação interativa.

1.2 Noções de Grafos

Um grafo $G = (V, E)$ é definido por um conjunto de vértices V e um conjunto de arestas $E \subseteq V \times V$. Representamos a aresta que liga um vértice p a um vértice q por (p, q). Dizemos que o grafo é orientado se cada aresta é um par ordenado. Caso contrário o grafo é dito não-orientado. As Figuras 1.1 (a–b) ilustram exemplos de grafo não orientado e orientado.

A cada vértice p de um grafo está associada uma lista de adjacências $A(p)$ que contém os vértices $q \in V$ tais que $(p, q) \in E$, ou seja, $A(p) = \{q \in V \ : \ (p, q) \in E\}$. No grafo da Figura 1.1(b), $A(5) = \{3, 4\}$. Dizemos que q é *adjacente* a p se $(p, q) \in E$. Um grafo $G' = (V', E')$ é um subgrafo de $G = (V, E)$ se $V' \subseteq V$ e $E' \subseteq E$.

Um caminho orientado (daqui em diante, simplesmente caminho) é uma seqüência de vértices $\langle v_1, v_2, \ldots, v_n \rangle$ sem repetições onde $v_{i+1} \in A(v_i)$, $\forall i \in [1, n-1]$. A Figura 1.2 (a) mostra um exemplo de caminho orientado. Cada caminho pode ter um custo associado,



(a) (b) (c)

Figura 1.1: Grafo não orientado (a) e orientado (b); (c) é um subgrafo de (b).

definido por uma função de custo caminho, a ser definida de acordo com a aplicação.

Dois vértices p e q são conexos se o grafo contém pelo menos um caminho não orientado de $P = \langle p, \ldots, q \rangle$. Um grafo é conexo se todo par de vértices $(p, q) \in V \times V$ for conexo.

Um ciclo orientado é formado por um caminho $\langle v_1, v_2, \ldots, v_n \rangle$ unido à aresta (v_n, v_1). Um grafo é acíclico se ele não contém nenhum ciclo orientado.

Chamamos um grafo acíclico e conexo de árvore. Definimos a raiz de uma árvore como sendo um vértice especial desta, que pode ser visto como o vértice a partir do qual a árvore "cresce". Uma árvore $T = (V, E') \subseteq G = (V, E)$ é dita de custos mínimos se o caminho (único) em T saindo da raiz de T para cada vértice $p \in V$ for um caminho de custo mínimo em G.

Uma floresta é um grafo acíclico, formado por uma ou mais árvores (subgrafos conexos) (Figura 1.2(b)). Dizemos que uma floresta é de caminhos mínimos se suas componentes conexas são árvores de caminhos de custo mínimo. Maiores detalhes sobre grafos, algoritmos em grafos e suas aplicações podem ser encontrados em [2], [10] e [38].

1.3 Noções de Imagens Digitais

Uma imagem n-dimensional com m bandas é representada por um conjunto de *spels* (do inglês *space elements*), que são subdivisões do $\mathbb{R}^n$ formando o domínio de imagem em $\mathbb{Z}^n$, e pela associação de m valores a cada spel. No caso 2D, os spels são chamados *pixels* (do inglês *picture elements*) e no caso 3D são chamados *voxels* (do inglês *volume elements*).

Imagens em escala de cinza associam a cada spel um valor numérico em um domínio totalmente ordenado, em geral um subconjunto de $\mathbb{Z}$; Imagens coloridas associam a cada spel 3 componentes de cor, em geral componentes vermelha (R), verde (G) e azul (B). Imagens de satélite associam um número maior de valores a cada spel, cada valor refere-se a

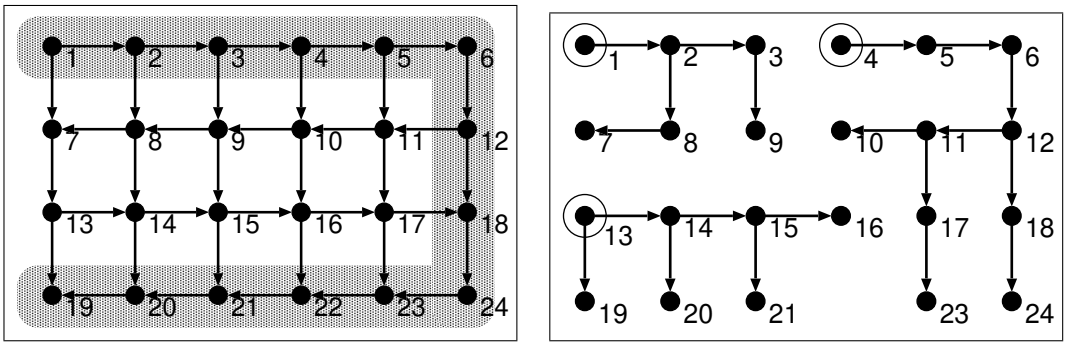


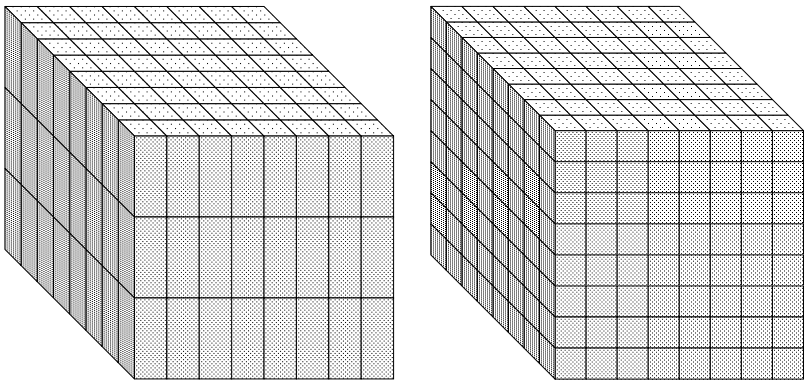
Figura 1.2: (a) Caminho orientado; (b) Uma floresta com três árvores, com raízes nos vértices 1, 4 e 13.

um tipo comprimento de onda (infra-vermelho, ultra-violeta, termal, etc.). Denominamos *volume* uma imagem tri-dimensional em que as três dimensões representam dimensões espaciais (vídeos têm 3 dimensões, porém 2 espaciais e 1 temporal).

Volumes de ressonância magnética (MR) e de tomografia computadorizada (CT) associam a cada voxel p apenas um valor, que denominaremos de intensidade do voxel, $I(p)$. Neste caso, o volume é definido por um conjunto finito de voxels $\mathcal{I} \subset \mathbb{Z}^3$ e um mapa de intensidades I. O domínio das intensidades (resolução radiométrica) é limitado pela sensibilidade do método de aquisição e, posteriormente, pelo espaço de armazenamento requerido para manter o mapa de intensidades I. Usa-se a notação *imagem de b bits* para designar uma imagem cujas intensidades são inteiros representáveis com b bits, ou seja, $0 \leq I(p) \leq 2^b - 1,\ \forall p \in \mathcal{I}$. São usadas imagens de 16 bits para representar a maioria das imagens médicas, embora os equipamentos de aquisição estejam limitados a domínios um pouco menores (imagens de 8 e 12 bits, por exemplo). A expressão *tamanho da imagem* refere-se ao número de spels que a compõem, ou seja, $|\mathcal{I}|$ (resolução espacial).

Uma característica a ser notada em volumes de MR e CT é que o processo de aquisição – a aquisição de uma fatia 2D por vez – gera voxels que representam paralelepípedos (em vez de cubos) do espaço real. A distância entre fatias consecutivas é, em geral, maior que a distância entre dois pixels adjacentes em uma mesma fatia. Para permitir a correta visualização dos dados, é comum calcular um novo volume com mais fatias (a partir de interpolação dos dados originais), em que cada voxel representa um cubo no espaço original. Este volume, formado por voxels cúbicos, é chamado volume *isotrópico*. É importante que a distância entre fatias não cause artefatos de sub-amostragem em regiões de topologia complexa, ou métodos 3D terão dificuldades devido à descontinuidade entre fatias. Em imagens neurológicas a descontinuidade passa a causar problemas quando a

distância entre fatias passa de 2,5 mm, mas este valor será diferente para cada órgão ou parte do corpo estudada.



(a) (b)

Figura 1.3: (a) volume anisotrópico; (b) volume isotrópico.

.

Capítulo 2

A transformada imagem-floresta

A transformada imagem-floresta [21] (IFT, do inglês Image Foresting Transform) é uma ferramenta genérica para o desenvolvimento de operadores de processamento de imagens. A IFT computa, de forma robusta e eficiente, uma floresta de custo mínimo em um grafo a partir de um conjunto de *spels* candidatos a raízes de árvores. Diferentes operadores de processamento de imagens são obtidos através da escolha apropriada de um mapeamento da imagem para um grafo e de uma função de custo de caminho [11, 18, 19, 20].

2.1 Mapeamento da Imagem em um Grafo

A forma mais natural de obter um grafo a partir de uma imagem é fazer de cada spel um vértice, e definir arestas entre spels que possuam algum tipo de adjacência, em qualquer sentido do termo. Mapeamentos genéricos, que utilizam uma "máscara" única para gerar a adjacência de cada vértice, permitem grande economia de espaço de armazenamento para o grafo, ocupando espaço $\Theta(|V|)$ para armazenar o grafo, em vez de $\Theta(|V| + |E|)$ que seria necessário com representação explícita das arestas.

Um tipo de *relação de adjacência* razoável é a adjacência Euclidiana (Figuras 2.1(a)–(d)), que define arestas saindo de um vértice para todos os outros vértices distantes até um dado raio (de acordo com a métrica Euclidiana). Este tipo de adjacência resulta em um grafo que pode ser percorrido com uma trajetória contínua no espaço, representando bem a propagação de diversos fenômenos físicos que desejamos simular no espaço da imagem, como alagamento por um fluido, propagação de uma chama ou difusão através de membranas. A Figura 2.1 mostra alguns exemplos de adjacência Euclidiana em imagens 2D e 3D.

Assim, o conjunto de spels $\mathcal{I}$ de uma imagem e uma relação de adjacência $\mathcal{A}$ definem um grafo $G = (\mathcal{I}, \{\mathcal{A}(p) \ : \ \forall p \in \mathcal{I}\})$ (Figura 2.1(e)).

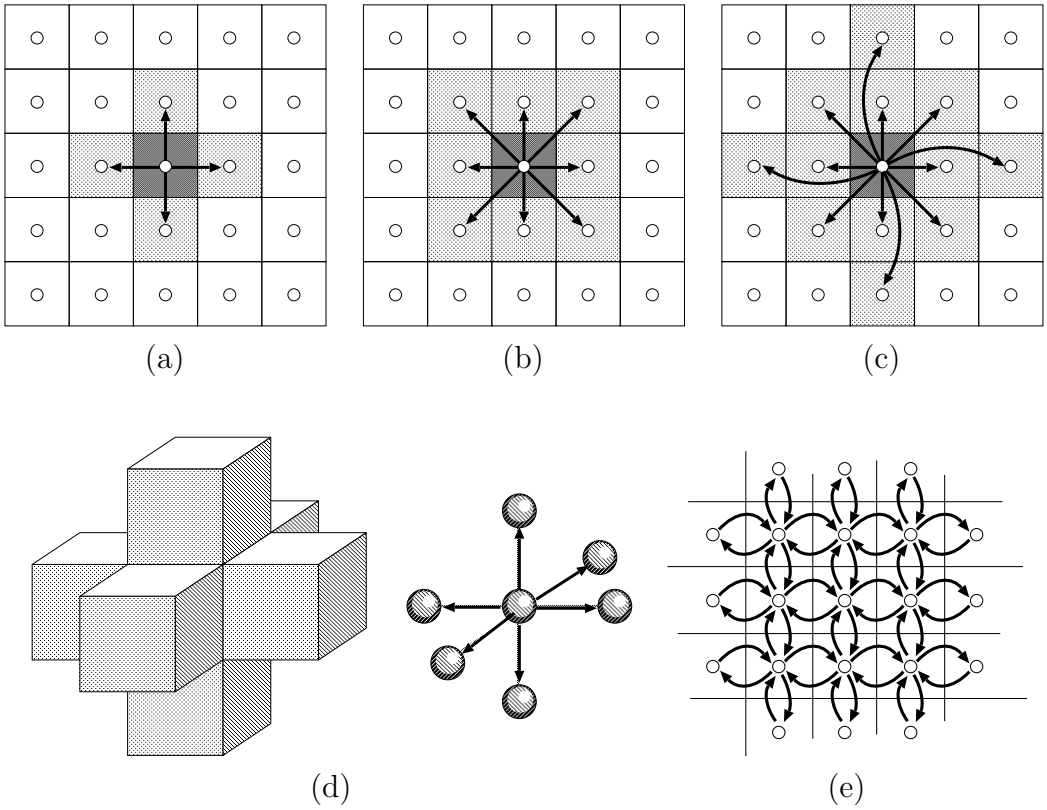


Figura 2.1: Adjacências Euclidianas: (a) 2D de raio 1, (b) 2D de raio $\sqrt{2}$, (c) 2D de raio 2, (d) 3D de raio 1, e (e) grafo resultante da adjacência (a).

2.2 Algoritmo da IFT

A IFT trabalha sobre uma estrutura de dados que denominamos *cena anotada*. Esta estrutura armazena os mapas de custos, raízes e predecessores da floresta associada à imagem $\mathcal{I}$:

Cena Anotada $\mathcal{S}$	**é**	
	P	: mapa de predecessores ($\mathcal{I} \rightarrow \mathcal{I} \cup \{nil\}$)
	R	: mapa de raízes ($\mathcal{I} \rightarrow \mathcal{I}$)
	C	: mapa de custos ótimos ($\mathcal{I} \rightarrow \mathbb{R}$)
fim.		

As arestas do grafo são dadas por uma relação de adjacência $\mathcal{A}$, e seus pesos são definidos implicitamente por uma função de custo de caminho f.

O mapa de predecessores indica qual é o spel $P(p)$ predecessor de cada spel p na floresta de caminhos mínimos, ou armazena um marcador *nil* para indicar que aquele spel é a raiz de uma árvore. O mapa de raízes indica qual é o spel raiz $R(p)$ da árvore de caminhos mínimos a que cada spel p pertence. O mapa de custos C armazena, para cada spel p, o custo $C(p)$ do caminho mínimo $\langle R(p), \ldots, p\rangle$.

A função f determina os custos de caminhos sobre o grafo, e a IFT exige que f seja *suave*, conforme a definição abaixo.

Definição 1 *(Função Suave) Seja $\tau = \langle R(p), \ldots, p\rangle$ e $\pi = \tau \cdot \langle p, q\rangle$, onde τ é um caminho de custo mínimo entre $R(p)$ e p. Uma função de custo de caminho f é dita suave se satisfizer as condições (C1)–(C3) abaixo para qualquer caminho π em seu domínio.*

(C1) $f(\tau) \leq f(\pi)$

(C2) τ *é de custo mínimo*

(C3) $f(\tau' \cdot \langle p, q\rangle) = f(\pi)$ *para todo caminho de custo mínimo* $\tau' = \langle \ldots, p\rangle$.

Os parâmetros $\mathcal{A}$ e f, juntos, condicionam a otimalidade da floresta, já que $\mathcal{A}$ determina as arestas que podem ser usadas para a formação de caminhos e f determina os custos de todos os possíveis caminhos. Não faz sentido dizer que uma floresta é ou não é ótima sem associar esta afirmação a um **contexto de otimalidade** $\{\mathcal{A}, f\}$.

Embora os valores em C e f possam ser Reais, usamos valores inteiros para permitir uma implementação da IFT em tempo linear, $O(|\mathcal{I}|)$. Na prática, os custos de caminhos são obtidos por funções simples (máximo ou soma, por exemplo) sobre as intensidades dos spels, que são também valores inteiros. É natural, portanto, o uso de funções f com valores em $\mathbb{Z}$, sem perda de informação.

A IFT admite, opcionalmente, um conjunto de sementes $\mathcal{M} \subseteq \mathcal{I}$ como parâmetro, restringindo o conjunto de raízes da floresta a $\mathcal{M}$. Note, entretanto, que não há garantia de que todos os spels de $\mathcal{M}$ se tornarão raízes de árvore: se $\{p, q\} \subseteq \mathcal{M}$ e $f(\langle q\rangle) > f(\langle p, q\rangle)$, o caminho de custo mínimo associado a q será $\langle p, q\rangle$, e q não será raiz de árvore. Em aplicações de segmentação de imagens, $\mathcal{M}$ tem grande importância e é composto for spels representativos dos objetos a serem segmentados, com $\mathcal{M} \ll \mathcal{I}$. Em geral, é desejável que o conjunto de raízes de árvores da floresta de caminhos mínimos seja um subconjunto de $\mathcal{M}$. Para garantir que isto ocorra, devemos escolher uma função de custo de caminho f restrita a valores finitos, garantindo que todos os spels da imagem serão conquistados por um caminho de custo finito partindo de alguma semente em $\mathcal{M}$.

O algoritmo da IFT é uma generalização do algoritmo de Dijkstra [14] para a solução do problema SSSP (Single-Source Shortest Paths, caminhos mais curtos a partir de uma única fonte), extendendo-o para permitir que diversas fontes (sementes) possam competir

por caminhos mínimos simultaneamente, resultando em uma partição ótima do grafo, com funções mais gerais de custo (funções suaves). A partição é ótima à medida em que garante que os spels dominados por uma raiz estão mais próximos (em algum sentido, dado pela escolha adequada de f) desta raiz do que de qualquer outra raiz de árvore da floresta resultante.

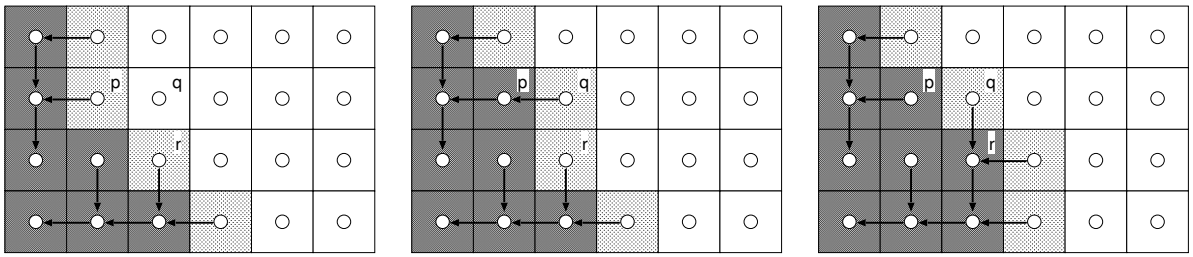
Algoritmo 2-1 – IFT

ENTRADA:	Imagem $\mathcal{I}$, Cena Anotada $\mathcal{S} = \{P, R, C\}$, Relação de Adjacência $\mathcal{A}$, Função de custo de caminho f, Conjunto de sementes $\mathcal{M} \subseteq \mathcal{I}$.
SAÍDA:	Cena Anotada $\mathcal{S}$ com floresta de caminhos mínimos.
AUXILIARES:	Fila de prioridades Q.

1. $Q \leftarrow \emptyset$
2. **Para Cada** $p \in \mathcal{I}$ **Faça**
3. $\quad P(p) \leftarrow nil,\ R(p) \leftarrow p,\ C(p) \leftarrow +\infty$
4. **Para Cada** $p \in \mathcal{M}$ **Faça**
5. $\quad R(p) \leftarrow p,\ C(p) \leftarrow f(\langle p \rangle)$
6. $\quad$ *Insira* p *em* Q *com prioridade* $C(p)$
7. **Enquanto** $Q \neq \emptyset$ **Faça**
8. $\quad$ *Remova de* Q *um spel* p *tal que* $C(p) \leq C(q)\ \forall q \in Q$
9. $\quad$ **Para Cada** *spel* $q \in \mathcal{A}(p)$ *tal que* $C(q) > C(p)$ **Faça**
10. $\quad\quad$ *Compute* $c' \leftarrow f(\langle R(p), \ldots, p \rangle \cdot \langle p, q \rangle)$
11. $\quad\quad$ **Se** $c' < C(q)$ **Então**
12. $\quad\quad\quad$ **Se** $q \in Q$ **Então**
13. $\quad\quad\quad\quad$ *Altere a prioridade de* q *em* Q *para* c'
14. $\quad\quad\quad$ **Senão**
15. $\quad\quad\quad\quad$ *Insira* q *em* Q *com custo* c'
16. $\quad\quad\quad P(q) \leftarrow p,\ C(q) \leftarrow c',\ R(q) \leftarrow R(p)$
17. **Retorne** $\mathcal{S} = \{P, R, C\}$

O algoritmo inicializa a cena com uma **floresta trivial** – todos os spels formam árvores triviais com custos infinitos (na prática, é usado o maior valor representável pela variável) – (linhas 2–3). Em seguida, os spels *sementes* são inicializados como raízes de árvores triviais e inseridos na fila de prioridades Q (linhas 4–6). Todo spel p presente na fila de prioridades é a extremidade de um caminho mínimo $\langle R(p), \ldots, p \rangle$. O laço das linhas 7–16 propaga uma frente de onda de caminhos mínimos que dominará todos os spels conexos do grafo aos quais consiga oferecer um custo de caminho menor que o atual. Como todos os spels fora de $\mathcal{M}$ são inicializados com custo $+\infty$, todos os spels conexos

serão conquistados[1]. A propagação remove um spel p por vez da fila de priopridades (linha 8), e tenta dominar os spels adjacentes, $q \in \mathcal{A}(p)$ (laço das linhas 9–16). Se a conquista for possível, por oferecer um caminho mínimo $\langle R(p), \ldots, p, q\rangle$ com custo menor que $C(q)$, o spel q é atualizado (predecessor, raiz, custo) e, caso não esteja na fila de prioridades, é inserido, pois agora faz parte da frente de propagação. O spel q pode já estar na fila em situações de "colisão" da frente de onda, como ilustrado na Figura 2.2. A função f precisa ser suave para garantir que os spels que já saíram da fila não precisam ser processados novamente. Um exemplo de função de custo não suave é ilustrado na Figura 2.3: f_{abs}, proposta por Adams e Bischof [1], que define o custo de um caminho $\pi = \tau \cdot \langle p, q\rangle$, com $\tau = \langle \ldots, p\rangle$, como o valor absoluto da diferença entre a intensidade de q e a média das intensidades dos spels no prefixo τ. Esta função não é suave porque não garante que prefixos de caminhos de custo mínimo sejam também caminhos de custo mínimo, sendo portanto impossível associar um predecessor único a cada spel.



(a) (b) (c)

Figura 2.2: Exemplo de situação de colisão na frente de onda. (a) Situação inicial (apenas os caminhos relevantes estão indicados); (b) p sai da fila, domina q e o insere na fila; (c) r sai da fila e domina q, que já estava na fila. (hachura escura: spel já saiu da fila; hachura intermediária: spel está na fila; fundo branco: spel ainda não entrou na fila.)

Funções de custo suave garantem que cada spel q será visitado apenas uma vez por cada spel p em que $q \in \mathcal{A}(p)$, portanto o algoritmo executa no máximo $|\mathcal{A}| \cdot |\mathcal{I}|$ iterações do laço entre as linhas 7–16.

Com relações de adjacência que levem a grafos esparsos ($|E| \ll |V|^2$), $|\mathcal{A}|$ pode ser considerada uma constante pequena e desconsiderada na análise assintótica.

Todos os spels são inseridos na fila pelo menos uma vez, e spels já removidos da fila (linha 8) nunca serão inseridos novamente, fixando o número de iterações do laço 7–16 em $|\mathcal{I}|$. Assim, podemos dizer que o algoritmo da IFT leva tempo $\Theta(|\mathcal{I}|)$ para ser executado. Este tempo é atingido com uma implementação de fila de prioridades em que o tempo das

[1]Desde que $f(\pi) < +\infty, \forall \pi$, conforme comentado anteriormente.

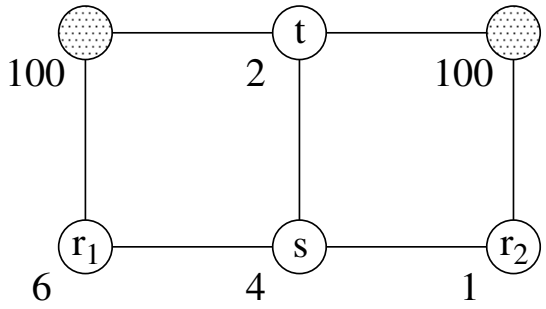


Figura 2.3: Exemplo de função de custo de caminho não-suave (f_{abs}, ver texto), em que a IFT não é aplicável: $f_{abs}(\langle r_1, s\rangle) = 2 < f_{abs}(\langle r_2, s\rangle) = 3$, mas $f_{abs}(\langle r_2, s, t\rangle) = 3 < f_{abs}(\langle r_1, s, t\rangle) = 4$, sendo impossível determinar um predecessor único para s que forme uma floresta de caminhos mínimos.

operações *Insere*, *Remove-Mínimo*, *Altera-Prioridade* e *Verifica-Presença* não depende de $|\mathcal{I}|$. Esta implementação está descrita no Apêndice A.

Uma descrição mais completa da IFT com provas de corretude e exemplos de aplicação pode ser encontrada em em [21]. [11, 36, 37, 49] descrevem operadores de processamento de imagens (2D) baseados na IFT.

Capítulo 3

A IFT diferencial

3.1 Definição da IFT diferencial

Na segmentação de imagens, as sementes são spels representativos dos objetos de interesse. A segmentação com correções interativas – em que o usuário muda o conjunto de sementes da IFT – depende da atualização da floresta de custos mínimos de acordo com o novo conjunto de sementes. Com o algoritmo da IFT, a única forma de obter tal atualização é recalcular a IFT em toda a imagem. A IFT diferencial (DIFT, do inglês *Differential IFT*) foi desenvolvida para resolver o problema da segmentação interativa de forma mais eficiente. Embora este trabalho aborde a segmentação semi-automática, em que as sementes são escolhidas pelo usuário, a atualização diferencial do conjunto de sementes é útil também para ajustar um conjunto de sementes escolhido automaticamente que não resultou em uma segmentação adequada.

A DIFT permite que o conjunto de raízes da floresta de caminhos mínimos seja modificado com a adição de novas sementes (que são candidatas a se tornarem raízes de árvores de caminhos mínimos) e remoção de árvores.

A DIFT tem como entrada a imagem $\mathcal{I}$, uma cena anotada $\mathcal{S}$, uma relação de adjacência $\mathcal{A}$, uma função suave e finita[1] de custo de caminho f (ver definições no Capítulo 2) e dois conjuntos de spels, $\mathcal{M}_I$ e $\mathcal{M}_R$. $\mathcal{M}_I$ é o conjunto de sementes a serem propagadas (e portanto candidatas a se tornarem raízes de novas árvores de caminhos mínimos), e $\mathcal{M}_R$ é um conjunto de spels cujas árvores serão removidas da floresta, denominados marcadores de remoção. A cena $\mathcal{S}$ deve conter uma floresta de caminhos mínimos válida ou uma floresta trivial[2]. O conjunto $\mathcal{M}_R$ define, implicitamente, um conjunto de raízes $\mathcal{R}_{MR}$ de árvores marcadas para remoção tal que

[1] $f(\pi) < +\infty, \forall \pi$.

[2] Em uma floresta trivial todos os spels formam árvores triviais com custos infinitos, $C(p) = +\infty$, $R(p) = p$ e $P(p) = nil \ \ \forall p \in \mathcal{I}$

$$p \in \mathcal{M}_R \Longrightarrow R(p) \in \mathcal{R}_{MR} \tag{3.1}$$

Definição 2 *Dada uma floresta trivial ou de caminhos mínimos (em relação a um contexto de otimalidade*[3] *$\{\mathcal{A}, f\}$, dado), representada por uma cena anotada $\mathcal{S}^{(0)}$, o conjunto $\mathcal{M}^{(0)}$ de raízes de $\mathcal{S}^{(0)}$, um conjunto de sementes $\mathcal{M}_I$ e um conjunto $\mathcal{R}_{MR} \subseteq \mathcal{M}^{(0)}$ de raízes de árvores marcadas para remoção (indicado implicitamente por um conjunto de marcadores $\mathcal{M}_R$, conforme a relação 3.1), a DIFT calcula uma floresta de caminhos mínimos $\mathcal{S}^{(1)}$ válida no mesmo contexto de otimalidade associado a $\mathcal{S}^{(0)}$ para o conjunto de sementes $\mathcal{M}^{(1)} = \left(\mathcal{M}^{(0)} \setminus \mathcal{R}_{MR}\right) \cup \mathcal{M}_I$.*

Note que a IFT passa a ser um caso particular da DIFT:

Corolário 1 *Dados uma floresta trivial $\mathcal{S}^{(0)}$, um conjunto de sementes $\mathcal{M}_I$ e um conjunto de marcadores de remoção $\mathcal{M}_R = \emptyset$, a DIFT calcula uma floresta de caminhos mínimos válida para o conjunto de sementes $\mathcal{M}_I$.*

Os custos infinitos garantem que as sementes em $\mathcal{M}_I$ dominarão todos os spels atingíveis, pois oferecerão custos menores a todos os spels que visitarem.

A aplicação de uma sequência de DIFTs com conjuntos de sementes $\mathcal{M}_I^{(1)}, \mathcal{M}_I^{(2)}, \ldots, \mathcal{M}_I^{(n)}$ e conjuntos raízes de árvores a serem removidas $\mathcal{R}_{MR}^{(1)}, \mathcal{R}_{MR}^{(2)}, \ldots, \mathcal{R}_{MR}^{(n)}$ sobre uma floresta inicial $\mathcal{S}^{(0)}$ com raízes $\mathcal{M}^{(0)}$ resulta em uma floresta de caminhos mínimos $\mathcal{S}^{(n)}$ válida para o conjunto de sementes $\mathcal{M}^{(n)}$ resultante dado pela expansão da relação de recorrência da Equação 3.2.

$$\mathcal{M}^{(n)} = \left(\mathcal{M}^{(n-1)} \setminus \mathcal{R}_{MR}^{(n)}\right) \cup \mathcal{M}_I^{(n)} \tag{3.2}$$

A seguir discutimos a propagação de novas sementes em uma floresta de custos mínimos já estabelecida (Seção 3.2), a remoção de árvores de uma floresta de custos mínimos (Seção 3.3) e a realização simultânea destas duas operações (Seção 3.4).

3.2 Propagação de Novas Sementes

A propagação de novas sementes em uma floresta de caminhos mínimos não precisa visitar todos os spels da cena, mas apenas aqueles que:

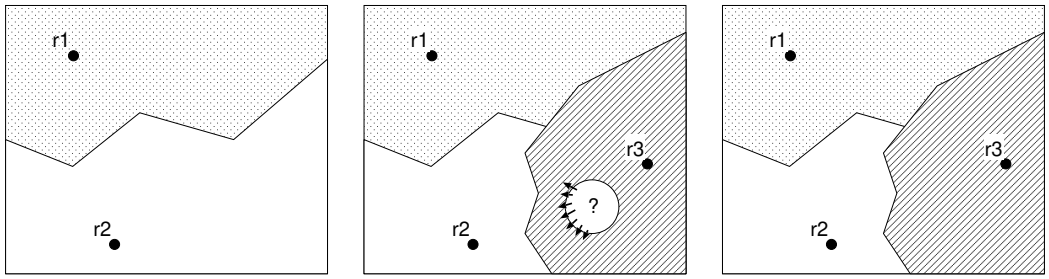
(1) Podem ser atingidos por caminhos enraizados em uma das novas sementes com custo menor ao custo oferecido por suas raízes na floresta pré-existente.

[3] *Ver seção 2.2*

(2) Tiveram seus caminhos mínimos modificados porque um de seus ancestrais no caminho mínimo na floresta pré-existente foi conquistado por uma das novas sementes. Neste caso, tanto a raiz como o custo do caminho associado podem ter mudado, e precisam ser recalculados em função do novo caminho mínimo.

A propagação de novas sementes pode ser realizada inserindo as sementes $\mathcal{M}_I$ na fila de prioridades, como na IFT, porém ao considerar os spels adjacentes a spels que saem da fila, um novo teste deve ser adicionado para garantir a visita e atualização dos spels enquadrados na condição (2) acima, já que o teste de custo da IFT apenas garante a visita aos spels enquadrados na condição (1).

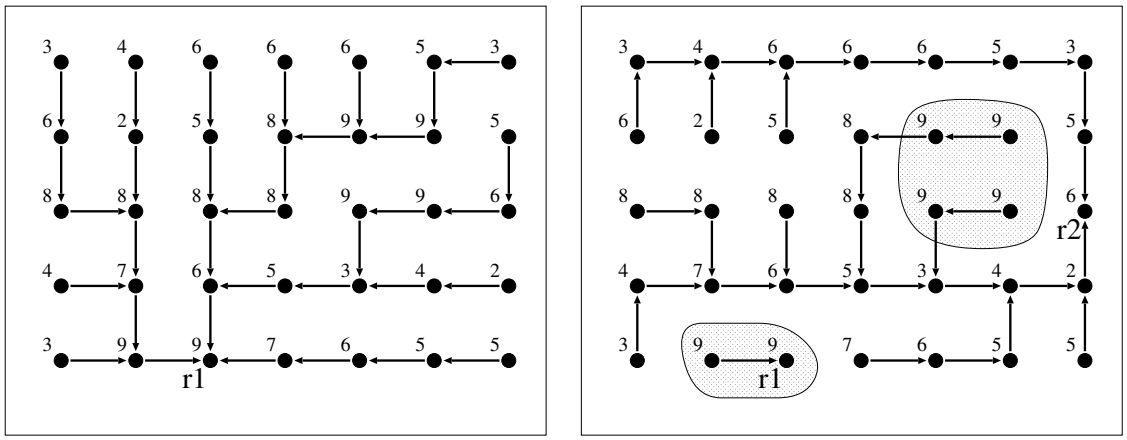
A condição (2) evita, principalmente, a formação de ilhas desconexas em regiões de empate entre uma raiz pré-existente e uma semente de $\mathcal{M}_I$. As Figuras 3.1 e 3.2 mostram exemplos desta situação.



(a) (b) (c)

Figura 3.1: Exemplo de formação de ilhas em áreas de empate. (a) Floresta inicial com duas árvores. (b) Propagação de nova semente $r3$ sem a conquista de spels que tiveram ancestrais modificados: a ilha tem raízes incorretas e custos possivelmente incorretos, portanto a cena anotada resultante é inválida porque o mapa R não é consistente com a floresta P. (c) Resultado correto, com visitação de spels que tiveram ancestrais modificados.

Note que uma dada raiz $p \in \mathcal{M}_I$ pode ter $f_0(p) > C(p)$, ou seja, um custo de caminho trivial que faça o caminho trivial custar mais que o caminho mínimo atual (enraizado em outro spel qualquer, $R(p) \neq p$). Neste caso, p deve ser ignorado (e não propagado), pois é uma semente *inviável* que não oferece custo ótimo sequer a si própria. Cada semente viável p tem seus atributos $C(p), R(p), P(p)$ modificados que p componha uma árvore trivial e tenha custo $f_0(p)$.



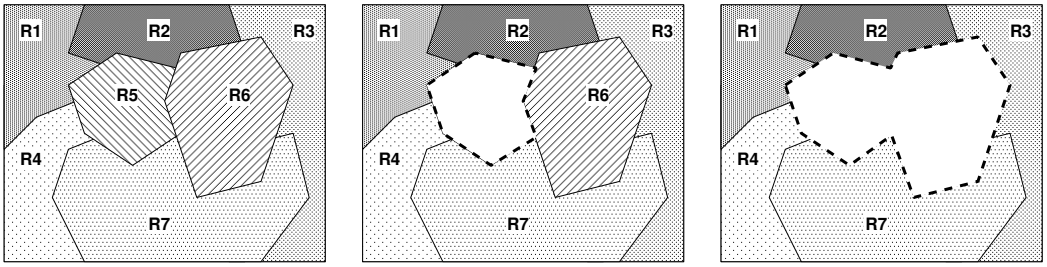
(a) (b)

Figura 3.2: Exemplo de formação de ilhas em áreas de empate, com a função de custo f_{max} (o custo do caminho é o maior valor de vértice presente no caminho). (a) Floresta inicial, com apenas uma raiz $r1$. (b) Floresta obtida com a adição da semente $r2$. Se a ilha de pixels com valor 9 (no topo, à direita) não for revisitada, o mapa de raízes R indicará erroneamente que seus pixels pertencem à árvore de $r1$.

3.3 Remoção de Árvores

A IFT diferencial deve remover árvores da floresta existente e calcular novos caminhos de custo mínimo para todos os spels que anteriormente possuíam seus caminhos mínimos enraizados na raiz da árvore removida. Para conquistar a região das árvores removidas é necessário computar um conjunto de fronteira $\mathcal{F}$ com todos os spels cujas raízes podem oferecer um caminho viável de custo mínimo a algum spel na região removida e realizar uma nova propagação de frentes de onda usando os spels em $\mathcal{F}$ como pontos de partida. Observe a Figura 3.3.

Os spels pertencentes às árvores não removidas já pertencem a caminhos mínimos e suas raízes continuam presentes, portanto não terão seus caminhos alterados. A região removida poderá ser conquistada pelas raízes restantes apenas por extensões de caminhos pré-existentes (candidatos a prefixos de caminho de custo mínimo para os spels na região removida) que conectem os spels da região removida a essas raízes. Os spels terminadores destes candidatos a prefixos de custo mínimo formam o conjunto $\mathcal{F}$ – spels adjacentes à região removida que pertencem a árvores não removidas. Para garantir a conquista com partição ótima da região removida, basta alterar o estado de seus spels para árvores triviais com custos infinitos (para garantir que serão conquistados por alguma raiz) e inserir os spels de $\mathcal{F}$ na fila de prioridades (sem reiniciar seus atributos na cena anotada



(a) (b) (c)

Figura 3.3: Exemplo de área de propagação na remoção de árvores da floresta. (a) Floresta original; (b) Remoção de uma árvore ($R5$); A borda tracejada indica os spels em $\mathcal{F}$, cujas raízes competem pela região removida; (c) Remoção simultânea de duas árvores ($R5$ e $R6$); A borda tracejada indica os spels em $\mathcal{F}$, cujas raízes competem pela região removida.

$\mathcal{S}$) e realizar o processamento da fila da IFT.

Note que os spels das árvores não removidas que fazem fronteira com a região removida não são necessariamente folhas. A Figura 3.4 mostra um exemplo desta situação.

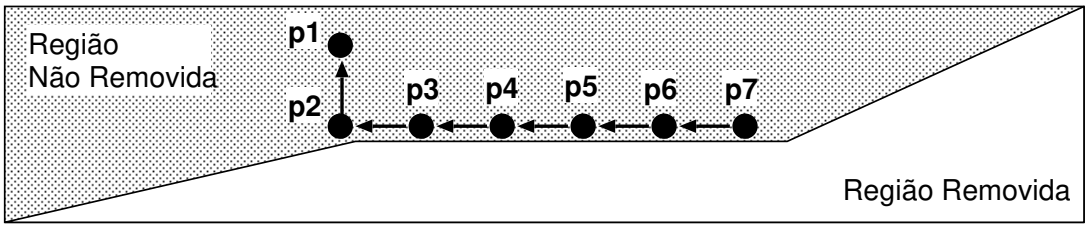


Figura 3.4: Exemplo de spels não-folhas na fronteira entre regiões removidas e não-removidas. $p2, \ldots, p7$ pertencem à fronteira $\mathcal{F}$, ainda que $p2, \ldots, p6$ não sejam folhas em suas árvores.

Os spels de fronteira podem ser encontrados realizando uma busca no grafo a partir de cada raiz de árvore removida, andando na "contramão" do mapa de predecessores P. A busca pára ao encontrar um spel pertencente a outra árvore que não aquela sendo removida. Se este spel que limita a busca pertencer a uma árvore não removida, então o mesmo é adicionado ao conjunto de spels de fronteira $\mathcal{F}$.

3.4 Combinando Adição de Sementes e Remoção de Árvores

É necessário tomar alguns cuidados ao combinar as operações de adição das sementes viáveis em $\mathcal{M}_I$ e remoção das árvores marcadas por $\mathcal{M}_R$. A fronteira $\mathcal{F}$ deve ser computada antes que as sementes em $\mathcal{M}_I$ sejam inicializadas, pois esta inicialização modifica os mapas P e R, o que pode tornar alguns spels inatingíveis para algortimo de busca de $\mathcal{F}$ (através da modificação do mapa P), e tornar incorretas decisões sobre a pertinência de spels a árvores removidas ou não removidas. A Figura 3.5 ilustra um exemplo de falha na busca caso a adição de sementes modifique os mapas P e R antes da realização da busca pela fronteira $\mathcal{F}$.

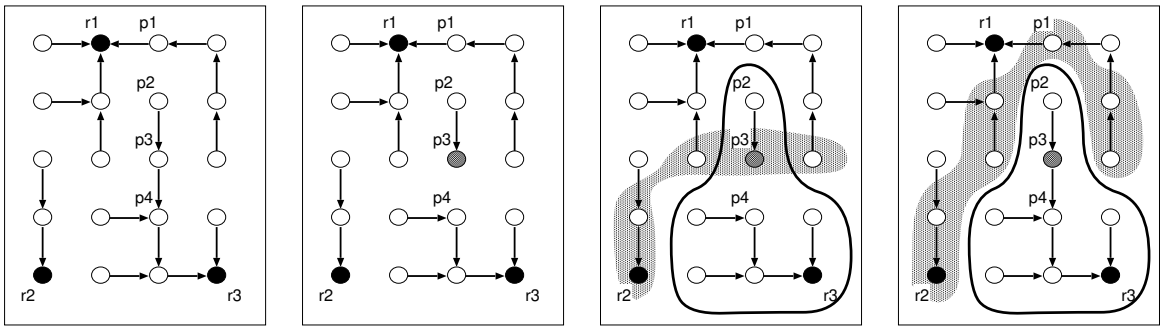


Figura 3.5: Combinação incorreta de adição e remoção: (a) Estado inicial da floresta. (b) $\mathcal{M}_R = \{r3\}$ e $\mathcal{M}_I = \{p3\}$. Neste exemplo incorreto de combinação, $P(p3)$ foi modificado antes da busca pela fronteira $\mathcal{F}$. (c) Fronteira $\mathcal{F}$ incorreta encontrada, devido à interrupção da busca quando $p3$ é visitado; (d) A Fronteira $\mathcal{F}$ correta que deveria ser encontrada na remoção da árvore de $r3$.

Quando uma semente viável $p \in \mathcal{M}_I$ coincide com um spel da fronteira $\mathcal{F}$ computada, devemos removê-lo de $\mathcal{F}$ e considerá-lo como semente, ou seja, seus atributos $C(p)$, $R(p)$ e $P(p)$ serão inicializados para uma árvore trivial com custo $f(\langle p \rangle)$. Este comportamento mantém a coerência da DIFT com a Definição 2. O tratamento de p como semente é a única forma de garantir que p apareça em $\mathcal{M}^{(n)}$, e portanto a única forma de garantir que uma seqüência de DIFTs obtenha uma floresta de caminhos mínimos para o conjunto de sementes $\mathcal{M}^{(n)}$ dado pela Equação 3.2.

3.5 Algoritmo da DIFT

Apresentamos a DIFT em dois algoritmos separados, por clareza. O procedimento auxiliar DIFT-LimpaArvores encontra as raízes dos spels no conjunto $\mathcal{M}_R$, constrói o conjunto fronteira $\mathcal{F}$ e inicializa a região removida com custos infinitos e árvores triviais.

O procedimento DIFT é a IFT diferencial propriamente dita.

Algoritmo 3-1 – DIFT

Entrada:	Imagem $\mathcal{I}$, Cena Anotada $\mathcal{S} = \{P, R, C\}$, Relação de Adjacência $\mathcal{A}$, Função suave de custo de caminho f, Conjunto de sementes a adicionar $\mathcal{M}_I \subset \mathcal{I}$, Conjunto de marcadores de árvores a remover $\mathcal{M}_R \subset \mathcal{I}$.
Saída:	Cena Anotada $\mathcal{S}$.
Auxiliares:	Fila de prioridades Q, Conjuntos $\mathcal{F}$ e $\mathcal{M}'_I$, mapa de cores $cor : \mathcal{I} \rightarrow \{branco, cinza, preto\}$.

1. $Q \leftarrow \emptyset$, $\mathcal{M}'_I \leftarrow \emptyset$
2. **Para Cada** $p \in \mathcal{I}$ **Faça** $cor(p) \leftarrow branco$
3. **Para Cada** $p \in \mathcal{M}_I$ **Faça**
4. ⌞ **Se** $f(\langle p \rangle) \leq C(p)$ **Então** $\mathcal{M}'_I \leftarrow \mathcal{M}'_I \cup \{p\}$
5. $(C, P, \mathcal{F}) \leftarrow$ DIFT-LimpaArvores$(C, R, P, \mathcal{A}, \mathcal{M}_R)$
6. $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{M}'_I$
7. **Para Cada** $p \in \mathcal{F}$ **Faça**
8. ⌞ *Insira* p *em* Q *com custo* $C(p)$, $cor(p) \leftarrow cinza$
9. **Para Cada** $p \in \mathcal{M}'_I$ **Faça**
10. $C(p) \leftarrow f(\langle p \rangle)$, $R(p) \leftarrow p$, $P(p) \leftarrow nil$
11. ⌞ *Insira* p *em* Q *com custo* $C(p)$, $cor(p) \leftarrow cinza$
12. **Enquanto** $Q \neq \emptyset$ **Faça**
13. *Remova de* Q *um spel* p *tal que* $C(p) \leq C(q)\ \forall q \in Q$
14. $cor(p) \leftarrow preto$
15. **Para Cada** *spel* $q \in \mathcal{A}(p)$ **Faça**
16. **Se** $cor(q) \neq preto$ **Então**
17. *Compute* $c' \leftarrow f(\langle R(p), \ldots, p \rangle \cdot \langle p, q \rangle)$
18. **Se** $c' < C(q)$ *ou* $P(q) = p$ **Então**
19. **Se** $q \in Q$ **Então**
20. ⌞ *Altere a prioridade de* q *em* Q *de* $C(q)$ *para* c'
21. **Senão**
22. *Insira* q *em* Q *com custo* c'
23. ⌞ $cor(q) \leftarrow cinza$
24. ⌞ $P(q) \leftarrow p$, $C(q) \leftarrow c'$, $R(q) \leftarrow R(p)$
25. **Retorne** $\mathcal{S} = \{P, R, C\}$

A DIFT usa um mapa de cores, *cor*, para indicar a relação de cada spel com a fila de prioridades. Spels *brancos* nunca entraram na fila, spels *cinza* estão na fila e spels *pretos* já saíram da fila. A suavidade da função de custo f garante que, uma vez que o spel sai da fila, seu caminho de custo mínimo já está determinado e não será alterado, portanto o spel não será inserido novamente na fila. A garantia de que cada spel entra no máximo uma vez na fila impõe um limite superior de $|\mathcal{I}|$ ao número de iterações do laço das linhas 12–24. O mapa de cores é inicializado na linha 2.

As linhas 3–4 constroem o conjunto $\mathcal{M}'_I$ de sementes viáveis, descartando os spels em $\mathcal{M}_I$ com custo de caminho trivial superior ao custo oferecido pela floresta de custos mínimos previamente estabelecida. Na primeira DIFT sobre uma cena, quando todos os spels têm custo $+\infty$, $\mathcal{M}'_I = \mathcal{M}_I$.

As linhas 5–6 constroem o conjunto fronteira $\mathcal{F}$, discutido na seção 3.3. $\mathcal{F}$ é construído pelo procedimento DIFT-LimpaArvores (Algoritmo 3-2, apresentado a seguir), e, conforme discutido na seção 3.4, spels na interseção entre $\mathcal{F}$ e $\mathcal{M}'_I$ são removidos de $\mathcal{F}$.

As linhas 7–11 inserem os spels de $\mathcal{F}$ e $\mathcal{M}'_I$ na fila de prioridades. Os spels sementes têm custo, predecessor e raiz inicializados (linha 10), mas os spels fronteira têm seus atributos atuais mantidos, pois representam suas árvores de caminhos mínimos na floresta previamente estabelecida. $\mathcal{M}'_I \cap \mathcal{F} = \emptyset$, portanto a ordem dos laços de inserção 7–8 e 9–11 não tem importância.

O laço principal de propagação da frente da IFT (linhas 12–24) é similar ao da IFT, com duas importantes modificações. O teste de predecessor $P(q) = p$ na linha 18 garante a atualização de spels que tiveram seus caminhos modificados nesta iteração da DIFT, conforme a condição (2) discutida na seção 3.2. O teste de cor $cor(q) \neq preto$ na linha 16 evita três computações inúteis: o cálculo de custo da linha 17 (que pode ser custoso, dependendo da função de custo f), a comparação $c' < C(q)$ na linha 18 e o teste de predecessor também na linha 18. Se o vizinho q sendo considerado já saiu da fila (preto), então seu caminho de custo mínimo já está calculado e pode ser descartado, evitando um cálculo e dois testes desnecessários. Devido ao uso do mapa *cor*, o teste de pertinência $q \in Q$ da linha 19 pode ser realizado verificando se $cor(q) = cinza$, evitando a necessidade de uma operação de verificação de pertinência explícita na implementação da fila Q.

A complexidade de tempo deste algoritmo é

$$T_{DIFT} = \left[\sum_{i=0}^{|\mathcal{M}_I|-1} \Gamma(\cup, i, 1) \right] + T_{LA} + \Gamma(\setminus, |\mathcal{F}|, |\mathcal{M}'_I|) + |\mathcal{F}| + |\mathcal{M}'_I| + (|\Psi| \cdot |\mathcal{A}|) \quad (3.3)$$

Onde $\Gamma(op, n, m)$ é o tempo necessário para realizar a operação op entre dois conjuntos com n e m elementos, respectivamente. A implementação dos conjuntos e a complexidade

de suas operações serão discutidos na seção A.2. T_{LA} é o limite superior de tempo do procedimento DIFT-LimpaArvores. Ψ é o conjunto de spels que precisam ser atualizados pela DIFT. $|\Psi| \leq |\mathcal{I}|$, e suas propriedades serão discutidas adiante.

A linha 2 do algoritmo pode ser implementada com operações de preenchimento de blocos com custo desprezível, por isso seu custo não ocorre na expressão. (Os seis termos que compõem T_{DIFT} correspondem, respectivamente, aos trechos 3–4, 5, 6, 7–8, 9–11 e 12–24 do Algoritmo 3-1).

Algoritmo 3-2 – DIFT-LimpaArvores

Entrada:	Mapa de Custos C, Mapa de Predecessores P, Mapa de Raízes R, Conjunto de marcadores de árvores a remover $\mathcal{M}_R \subset \mathcal{I}$, Relação de Adjacência $\mathcal{A}$
Saída:	C, P, $\mathcal{F}$.
Auxiliares:	Fila FIFO T, Conjunto $\mathcal{R}_{MR}$ de raízes de árvores a remover.

1. $\mathcal{R}_{MR} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$
2. **Para Cada** $p \in \mathcal{M}_R$ **Faça**
3. $\quad q \leftarrow R(p)$
4. $\quad \mathcal{R}_{MR} \leftarrow \mathcal{R}_{MR} \cup \{q\}$
5. $\quad$ **Se** $C(q) \neq +\infty$ **Então**
6. $\quad\quad$ *Insira q em T*, $C(q) \leftarrow +\infty$, $P(q) \leftarrow nil$
7. **Enquanto** T *não estiver vazia,* **Faça**
8. $\quad$ *Remova p de T*
9. $\quad$ **Para Cada** q *tal que* $(p, q) \in \mathcal{A}$, **Faça**
10. $\quad\quad$ **Se** $P(q) = p$ **Então**
11. $\quad\quad\quad$ $C(q) \leftarrow +\infty$, $P(q) \leftarrow nil$, *Insira q em T*
12. $\quad\quad$ **Senão** , **Se** $R(q) \notin \mathcal{R}_{MR}$ **Então** $\mathcal{F} \leftarrow \mathcal{F} \cup \{q\}$
13. **Retorne** $(C, P, \mathcal{F})$

As linhas 2–6 encontram as raízes das árvores marcadas por spels em $\mathcal{M}_R$, inicializam seus atributos como árvores triviais de custos infinitos, e as inserem na fila T. O teste da linha 5 evita que uma mesma raiz seja inserida duas vezes em T.

O laço das linhas 7–12 propaga as raízes das árvores a serem removidas com um percurso em largura. O propagação continua enquanto o vizinho q sendo considerado pertencer à mesma árvore a que pertence p. Caso q pertença a uma árvore não-removida (teste $R(q) \notin \mathcal{R}_{MR}$ na linha 13), q é colocado no conjunto fronteira $\mathcal{F}$, conforme discutido na seção 3.3. Durante a propagação, todos os spels pertencentes a árvores removidas são visitados e têm seus atributos inicializados de forma que sejam necessariamente visitados e conquistados por algum caminho na propagação da DIFT (árvores triviais com custos infinitos).

A complexidade de tempo deste algoritmo é

$$T_{LA} \leq \left[\sum_{i=0}^{|\mathcal{M}_R|-1} \Gamma(\cup, i, 1) \right] + \{|\Lambda| \cdot |\mathcal{A}| \cdot [\Gamma(\not\in, |\mathcal{M}_R|, 1) + \Gamma(\cup, |\mathcal{F}| - 1, 1)]\} \tag{3.4}$$

Onde Λ é o conjunto de spels atualmente pertencentes às árvores marcadas por $|\mathcal{M}_R|$, $|\Lambda| \leq |\mathcal{I}|$. Note que o termo $\Gamma(\cup, |\mathcal{F}| - 1, 1)$ é um limite superior, pois o conjunto terá $|\mathcal{F}| - 1$ elementos apenas na última operação de união (por isso a relação $\leq$ em vez de igualdade). Nossa implementação de conjunto (descrita no Apêndice A) realiza operações de união entre um conjunto e um elemento em $\Theta(1)$, portanto todo o termo será simplificado para 1 na análise final da complexidade da DIFT.

3.6 Complexidade da DIFT

Analisamos a seguir os requerimentos de tempo da DIFT, assumindo o uso da implementação da estrutura de dados para conjuntos descrita na Seção A.2.

Substituindo os tempos $\Gamma(\cup)$, $\Gamma(\not\in)$ e $\Gamma(\setminus)$ nas equações 3.3 e 3.4, temos

$$T_{DIFT} = |\mathcal{M}_I| + T_{LA} + 2(|\mathcal{F}| + |\mathcal{M}'_I|) + (|\Psi| \cdot |\mathcal{A}|) \tag{3.5}$$

$$T_{LA} = |\mathcal{M}_R| + |\Lambda| \cdot |\mathcal{A}| \tag{3.6}$$

Portanto, cada aplicação da DIFT consome tempo T_{DIFT} dado pela Equação 3.7.

$$T_{DIFT} = |\mathcal{M}_I| + |\mathcal{M}_R| + 2(|\mathcal{F}| + |\mathcal{M}'_I|) + |\mathcal{A}| \cdot (|\Psi| + |\Lambda|) \tag{3.7}$$

Ψ é o conjunto de spels que precisam ser atualizados pelo laço principal do Algoritmo 3-1, composto por:

(a) Spels que podem ser atingidos por uma semente em $\mathcal{M}_I$ com custo menor que na floresta anterior;

(b) Spels que possuem em seu caminho mínimo algum spel que se enquadre na condição (a) acima – em outras palavras, todos os descendentes dos spels em (a) na floresta de caminhos mínimos previamente estabelecida;

(c) Spels que nunca foram conquistados por um caminho mínimo;

(d) Spels na fronteira $\mathcal{F}$; e

(e) Spels pertencentes a árvores removidas (marcadas por $\mathcal{M}_R$).

Λ é o conjunto de spels visitados pela busca do Algoritmo 3-2, composto por $\mathcal{F}$ e pelos spels que se enquadram no item (e) acima, portanto

$$\Lambda \subset \Psi \tag{3.8}$$

Os tamanhos de Ψ e Λ são os componentes mais significativos no tempo da DIFT. Os outros conjuntos cujos tamanhos influem em T_{DIFT} são menores que Ψ:

$$(\mathcal{F} \cup \mathcal{M}_R) \subset \Lambda \subset \Psi \tag{3.9}$$

$$\mathcal{M}'_I \subset \Psi \tag{3.10}$$

Embora $\mathcal{M}_I$ possa ser muito maior que $\mathcal{M}'_I$, devido à inclusão de muitas sementes inviáveis, a viabilidade depende dos custos de caminhos triviais, definidos pela função de custo de caminho f, e em muitos operadores instanciados pela (D)IFT caminhos triviais têm custo 0, garantindo que $\mathcal{M}'_I = \mathcal{M}_I$, portanto a relação $\mathcal{M}_I \subset \Psi$ passa a ser válida.

O termo $|\Psi|+|\Lambda|$ é multiplicado pela tamanho da adjacência (quantas arestas saem de cada spel), mas em todas as aplicações práticas da (D)IFT usa-se relações de adjacência que levam a grafos esparsos, isto é, $|\mathcal{A}| \ll |\mathcal{I}|$, e podemos considerar que $|\mathcal{A}|$ é uma constante pequena. As relações de adjacência mais comuns são vizinhanças de raios 1, $\sqrt{2}$ (2D) ou $\sqrt{3}$ (3D) medidos em distância Euclidiana, excluindo auto-arestas. Em imagens 2D, a vizinhança de raio 1 tem $|\mathcal{A}| = 4$ e a de raio $\sqrt{2}$ tem $|\mathcal{A}| = 8$; Em imagens 3D, a vizinhança de raio 1 tem $|\mathcal{A}| = 6$ e a de raio $\sqrt{3}$ tem $|\mathcal{A}| = 26$.

Os tamanhos de Ψ e Λ claramente dependem da operação sendo realizada (conjuntos $\mathcal{M}_I$ e $\mathcal{M}_R$), e da topologia da floresta previamente estabelecida, que por sua vez depende dos atributos da cena anotada $\mathcal{S}$ e do contexto de otimalidade $\{\mathcal{A}, f\}$. Mesmo assim, podemos afirmar que:

(1) O conjunto Λ é composto de spels que necessariamente precisam ser inicializados com custos infinitos (regiões removidas) e por spels da região de fronteira $\mathcal{F}$ que necessariamente precisam ser detectados para compor a frente inicial de propagação da DIFT.

(2) O conjunto Ψ é composto de spels que necessariamente precisam ser visitados pela frente de propagação da DIFT para garantir a otimalidade da nova floresta.

Conforme argumentado nas seções 3.2–3.4. Portanto, embora $|\Psi| + |\Lambda|$ possa variar de forma pouco previsível no intervalo $[0, 2|\mathcal{I}|]$, nunca realizamos mais visitas do que o estritamente necessário.

Analisamos a seguir a performance em alguns casos particulares da DIFT. Assumimos sempre grafos esparsos com $|\mathcal{A}| \ll |\mathcal{I}|$.

Primeira DIFT

No caso da primeira aplicação da DIFT, ela equivale a uma IFT e tem o mesmo custo $\Theta(|\mathcal{I}|)$ da IFT (assumindo um grafo esparso, ou o custo seria $\Theta(|\mathcal{I}| \cdot |\mathcal{A}| \leq |\mathcal{I}|^2)$ tanto para a IFT como para DIFT): Todos os spels têm custo infinito e $\mathcal{M}_R = \emptyset$, portanto $\Psi = \mathcal{I}$ e $\Lambda = \emptyset$, e $T_{DIFT} = |\mathcal{M}_I| + 2|\mathcal{M}'_I| + |\mathcal{A}| \cdot |\mathcal{I}|$. $T_{DIFT} \in \Theta(|\mathcal{I}|)$, atingindo o mesmo custo assintótico da IFT.

Somente adição

Com $\mathcal{M}_R = \emptyset$, $\Lambda = \mathcal{F} = \emptyset$ e a DIFT apenas propaga as sementes viáveis $\mathcal{M}'_I$ obtidas de $\mathcal{M}_I$. Neste caso, $T_{DIFT} = |\mathcal{M}_I| + 2|\mathcal{M}'_I| + |\mathcal{A}| \cdot |\Psi|$. $T_{DIFT} \in \Theta(|\Psi|) \subset O(|\mathcal{I}|)$. O tempo depende da quantidade de spels que as novas sementes conseguem conquistar.

Somente remoção

Com $\mathcal{M}_I = \emptyset$, $\mathcal{M}'_I = \emptyset$, e $\Psi = \Lambda$. A DIFT leva tempo proporcional a $|\Lambda|$, que é o tamanho das árvores removidas. $T_{DIFT} = |\mathcal{M}_R| + 2|\mathcal{F}| + 2|\mathcal{A}| \cdot |\Lambda|$. $T_{DIFT} \in \Theta(|\Lambda|) \subset O(|\mathcal{I}|)$.

Pior caso

No pior caso, $\mathcal{M}_R = \mathcal{I}$ (todas as árvores serão removidas, com $\Lambda = \mathcal{I}$ e $\mathcal{M}_I \neq \emptyset$, com $\Psi = \mathcal{I}$. neste caso, $\mathcal{F} = \emptyset$ pois não há spels em regiões não removidas. $T_{DIFT} = 3|\mathcal{I}| + 2|\mathcal{A}| \cdot |\mathcal{I}|$ e $T_{DIFT} \in \Theta(|\mathcal{I}|)$.

Casos patológicos

Quando $\mathcal{M}_I = \mathcal{M}_R = \emptyset$, a DIFT deixa a floresta inalterada e leva tempo $\Theta(1)$.

Quando $\mathcal{M}_I = \emptyset$ e todas as árvores são removidas ($\Lambda = \mathcal{I}$), a floresta é devolvida ao estado indefinido inicial (todos os spels são árvores triviais com custos infinitos). Neste caso curioso, o fator $|\Psi|$ não aparece no custo de tempo porque embora $\Psi = \mathcal{I}$, nenhum spel é adicionado à fila de prioridades Q do Algoritmo 3-1 ($\mathcal{F} = \mathcal{M}'_I = \emptyset$) e nenhuma iteração do laço de propagação das linhas 12–24 é executada. $T_{DIFT} = |\mathcal{M}_R| + |\mathcal{A}| \cdot |\mathcal{I}|$, e $T_{DIFT} \in \Theta(|\mathcal{I}|)$. Embora esta seja uma forma válida de obter a floresta inicial, é possível obter esta floresta de forma mais eficiente inicializando todos os spels sem a necessidade de uma fila FIFO e sem analisar os vizinhos de cada spel.

Capítulo 4

Segmentação baseada em regiões

Segmentar uma imagem consiste em gerar um mapeamento entre elementos de um conjunto de objetos de interesse e os elementos da imagem (spels). Em geral, estamos interessados em mapeamentos que relacionem cada spel a exatamente um objeto. De forma mais formal, a segmentação é o particionamento da imagem em k objetos O_i, $i = 1, 2, \ldots, k$, tais que $\bigcup_{i=1}^{k} O_i = \mathcal{I}$ e $O_i \cap O_j = \emptyset$ para todo $(i, j) \in (\{1, 2, \ldots, k\} \times \{1, 2, \ldots, k\})$ com $i \neq j$.

As estratégias de segmentação podem ser divididas em 3 grupos principais: por regiões, por bordas e baseada em modelos. A abordagem por regiões realiza a classificação com base nas propriedades de todos os spels pertencentes ao interior dos objetos [1]. A abordagem por bordas [30, 39] tenta detectar as fronteiras entre os objetos, resultando em uma definição explícita das superfícies dos objetos. A abordagem baseada em modelos [9, 29, 40, 52] é aplicável apenas em imagens adquiridas em condições controladas, em que se possa realizar uma normalização do espaço de coordenadas. Neste caso, a classificação se baseia em um mapa de probabilidades estabelecido a partir da segmentação de outras imagens adquiridas da mesma forma (atlas), e a normalização espacial permite a aplicação do modelo sobre a nova imagem.

Cada estratégia tem vantagens e desvantagens, e é cada vez mais comum o desenvolvimento de métodos híbridos [32, 42]. Neste trabalho nos concentramos em métodos baseados em regiões, que têm implementação direta a partir da IFT para qualquer dimensão. Métodos baseados em regiões são também mais diretos e intuitivos que os métodos baseados em bordas, evitando algumas complicações relacionadas à representação das bordas e ao casamento desta representação com o espaço discreto da imagem. Métodos baseados em modelos são muito dependentes da aplicação (objetos a serem segmentados, modalidade de aquisição da imagem, entre outros fatores), e requerem a construção de um modelo estatístico especializado.

Neste trabalho usamos a IFT diferencial para implementar e avaliar dois operadores

de segmentação por regiões: *Watershed* e *Conexidade Fuzzy Relativa*, discutidos neste capítulo.

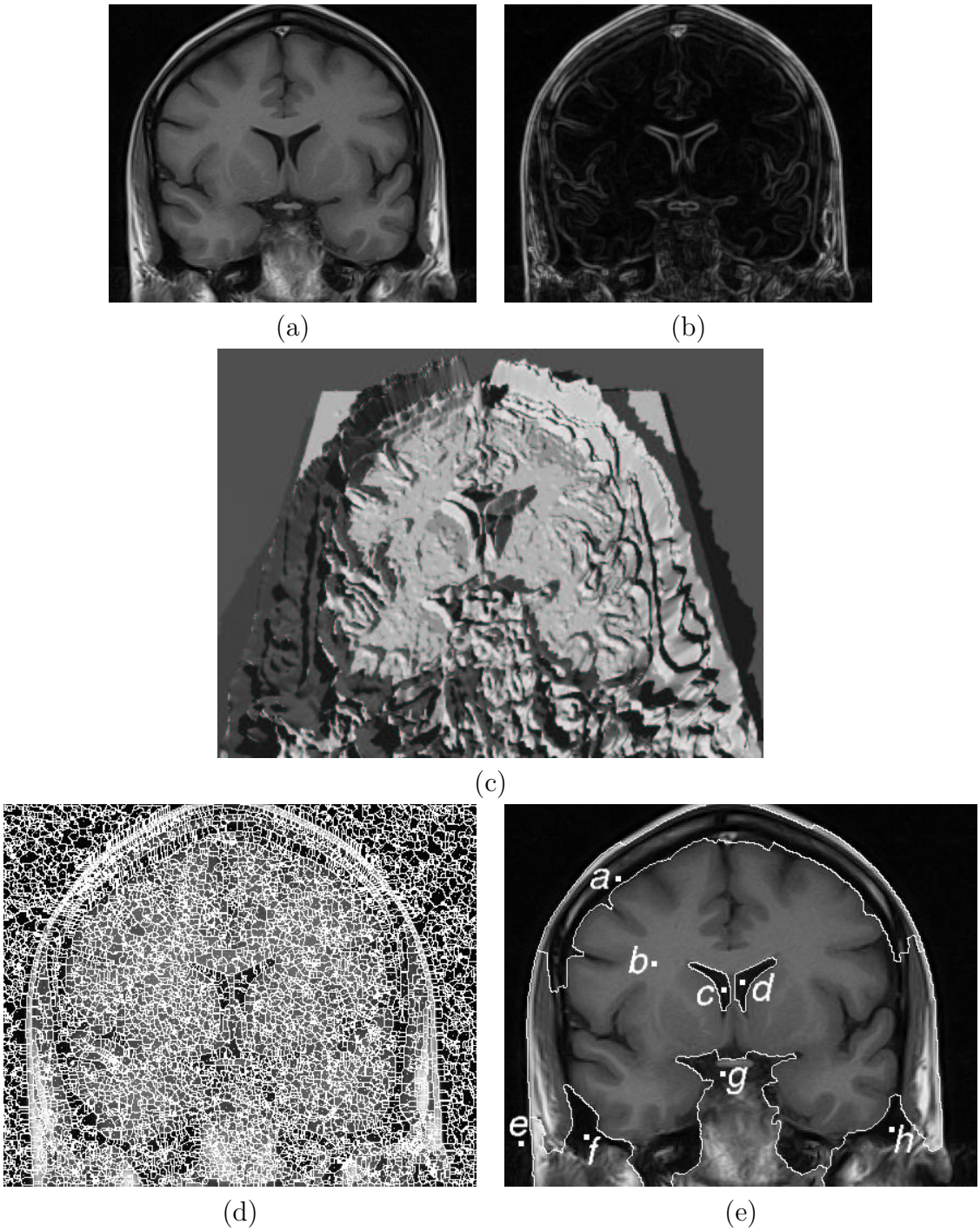
4.1 Watershed

A transformada de Watershed[1] foi proposta originalmente por Beucher e Meyer [6] e popularizada por Vincent e Soille [53]. Desde então vem sendo utilizada com sucesso para a segmentação de imagens [26, 36, 7, 23, 32, 37]. A formulação original da transformada de Watershed se baseia na simulação de imersão da imagem (com as intensidades dos spels representando alturas, e furos nos mínimos das bacias, permitindo a entrada da água) em um corpo d'água e a detecção dos eventos de *encontro de águas* provenientes de diferentes bacias, formando divisores naturais entre as bacias. Ao final do processo de imersão, cada mínimo terá definido sua região de influência. Esta formulação original, entretanto, gera uma supersegmentação da imagem. A transformada de watershed é calculada sobre uma imagem de gradiente da imagem original, onde as fronteiras entre objetos formam barreiras para o processo de alagamento, e os interiores dos objetos formam bacias.

Para obter a segmentação desejada, pode-se utilizar algum algoritmo para fundir as bacias encontradas pelo Watershed, ou então usar a variação *Watershed de marcadores*, em que a água é "despejada" apenas em pontos específicos (marcadores), definidos pela aplicação. Nesta variação, os furos para a entrada da água existem apenas nos marcadores, e realiza-se a detecção dos encontros de águas normalmente.

A Figura 4.1 ilustra o uso da transformada de watershed sobre uma imagem bidimensional de ressonância magnética. As bordas mostradas na Figura 4.1e são um subconjunto das bordas da Figura 4.1d, obtidas com frentes de água iniciadas apenas nos 8 marcadores a–h indicados. O watershed requer, entretanto, uma imagem de gradiente adequada, em que os interiores dos objetos formem bacias e as fronteiras entre objetos formem relevos. Podem haver relevos nos interiores dos objetos, desde que sejam mais baixos que os relevos nas fronteiras entre objetos. O cálculo deste gradiente é dependente da modalidade de aquisição da imagem, bem como dos objetos que desejamos segmentar. Os procedimentos de pré-processamento usados neste trabalho serão discutidos nas próximas seções.

[1] A trasformada de Watershed foi algumas vezes traduzida como transformada de Linhas Divisoras de Águas, mas neste trabalho utilizaremos o nome original em inglês.



(a) (b)

(c)

(d) (e)

Figura 4.1: (a) Imagem de ressonância magnética; (b) Imagem de gradiente de (a); (c) Visualização de (b) como um relevo, visto de cima; (d) Resultado da transformada de Watershed, sem marcadores; (e) Exemplo de watershed com marcadores.

4.1.1 IFT-Watershed

A transformada de Watershed pode ser eficientemente implementada com a IFT [36]. A função de custo de caminho para o Watershed de marcadores é apresentada na Equações 4.1–4.2.

$$f(\langle t \rangle) = 0 \tag{4.1}$$

$$f(\pi \cdot \langle s,t \rangle) = max(f(\pi), grad(t)) \tag{4.2}$$

onde $grad(t)$ é o valor de gradiente no spel t. A mesma função de custo de caminho pode ser usada no algoritmo DIFT (Capítulo 3) para obter o operador diferencial DIFT-Watershed, permitindo o ajuste dinâmico do conjunto de marcadores, baseado nos resultados obtidos com iterações da DIFT.

4.2 Conexidade Fuzzy Relativa

Métodos de análise de imagens por conexidade fuzzy foram introduzidos em 1996 [51] e desde então foram extendidos [45, 46] e usados em aplicações específicas na área de processamento de imagens médicas [41, 33].

A conexidade fuzzy se baseia no conceito de afinidade entre spel e objeto. Dado um spel s, com intensidade $I(s)$ e um objeto O_i composto por spels cujas intensidades têm média μ_i e desvio padrão σ_i, a afinidade $\alpha(s, O_i)$ entre o spel s e o objeto O_i é dada pela Equação 4.3, que gera uma resposta Gaussiana em torno da média μ_i, conforme ilustrado na Figura 4.2.

$$\alpha(s, O_i) = exp\left(\frac{-(I(s) - \mu_i)^2}{2\sigma_i^2}\right) \tag{4.3}$$

A conexidade fuzzy absoluta [51] computa as afinidades de todos os spels da imagem em relação a um ou mais objetos representados por um spel semente inicial. Em seguida, é realizada uma limiarização para que spels com afinidade abaixo de um valor limite sejam considerados pertencentes ao "fundo" (objeto nenhum). Nesta abordagem, toda vez que a afinidade de um novo spel é calculada, μ_i e σ_i precisam ser reavaliados, o que leva a um algoritmo lento, especialmente para imagens tridimensionais.

A conexidade fuzzy relativa [46] contorna este problema evitando a reavaliação de todo o objeto a cada spel processado. Apenas spels próximos (em algum sentido adequado, em geral orientado à implementação) são considerados para a reavaliação de μ_i e σ_i.

Neste trabalho usamos a DIFT para implementar um operador de conexidade fuzzy relativa simplificado — DIFT-CFR — para a segmentação de um número arbitrário de

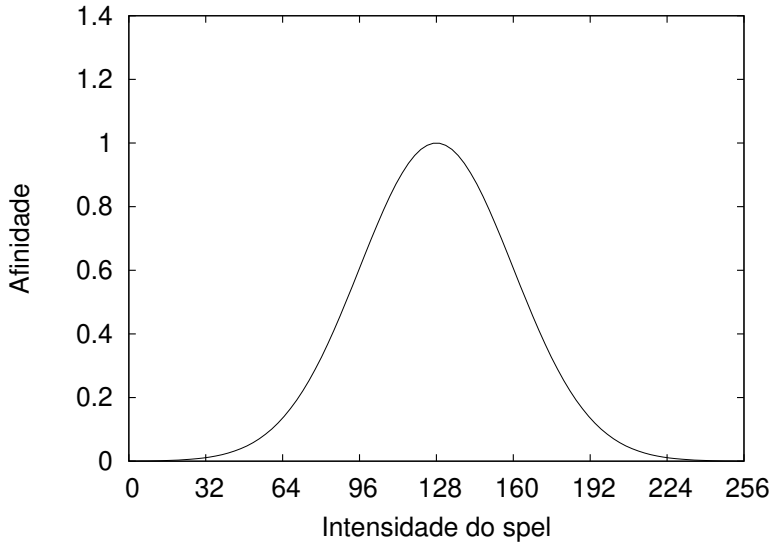


Figura 4.2: Gráfico da função de afinidade da Equação 4.3, para $\mu_i = 128$ e $\sigma_i = 32$.

objetos, em que μ_i e σ_i $(i = 1, 2, \ldots, k)$ são parâmetros de entrada do operador, representando média e desvio padrão esperados para cada objeto O_i a ser segmentado. O função de custo de caminho do operador DIFT-CFR é também um caso específico da função f_{max}, em que a afinidade de um spel em relação ao objeto a que pertence é a afinidade associada à aresta com menor afinidade (elo mais fraco) no caminho de custo mínimo que o conquista. A afinidade associada a cada aresta é a maior afinidade entre a média de intensidades nos spels da aresta e cada um dos k objetos. A função de custo de caminho do operador DIFT-CFR é apresentada nas Equações 4.4–4.6.

$$\alpha(\langle s,t\rangle, O_i) = exp\left(\frac{-\left(\frac{I(s)+I(t)}{2} - \mu_i\right)^2}{2\sigma_i^2}\right) \tag{4.4}$$

$$f(\langle t\rangle) = 0 \tag{4.5}$$

$$f(\pi \cdot \langle s,t\rangle) = K\left[1 - max_{i=1}^{k}\left(\alpha\left(\langle s,t\rangle, O_i\right)\right)\right] \tag{4.6}$$

A constante K é definida de acordo com a implementação, de forma a controlar o intervalo de valores de custos possíveis. Como a DIFT *minimiza* o custo do caminho, usamos o complemento da afinidade (que desejamos maximizar). Note que os k pares (μ_i, σ_i) precisam ser mantidos fixos entre iterações do DIFT-CFR sobre uma mesma floresta de caminhos mínimos: a alteração destes parâmetros exigiria a recomputação da

floresta de caminhos mínimos para toda a cena. O operador DIFT-CFR não usa qualquer operação de limiarização para exclusão do fundo, que deve ser tratado como um objeto adicional.

4.3 Pré-Processamento

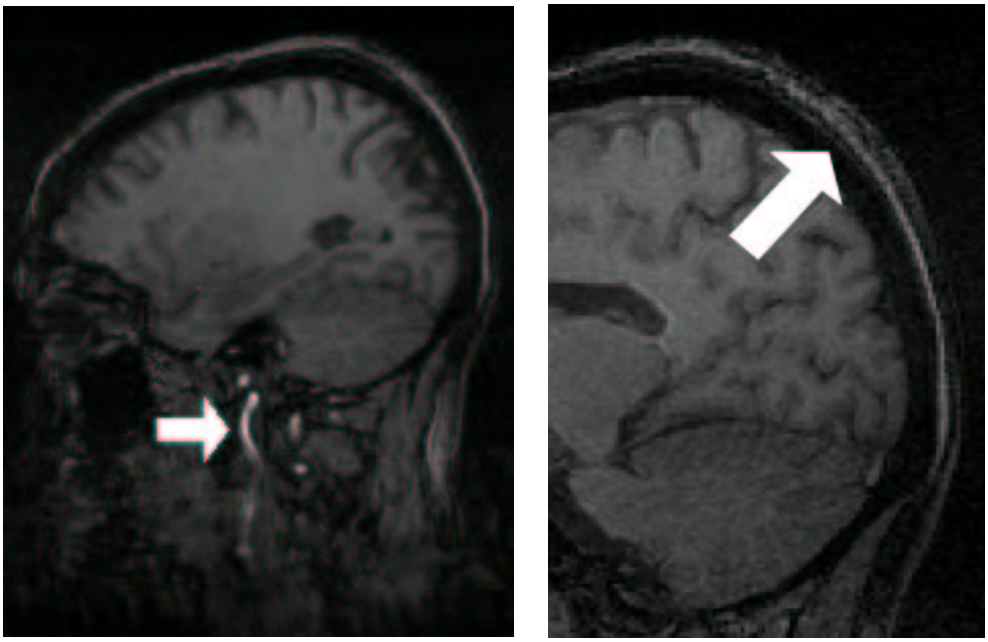
4.3.1 Watershed

A segmentação com watershed requer o cálculo de uma imagem de intensidades de gradientes a partir da imagem original. Isto gera um problema adicional de perda de resolução, uma vez que o gradiente precisa gerar bordas "fechadas" que evitem vazamentos. Por isso mesmo, é interessante usar a menor relação de adjacência possível – vizinhança 3D de raio 1 (Figura 2.1(d)) – pois o uso de relações de adjacência maiores requereriam operadores de gradiente mais "grossos", causando perda de resolução ainda maior, além de aumentar o tempo requerido para computar a DIFT. Dentre diversos algoritmos testados, o gradiente morfológico [25] com elemento estruturante $3 \times 3 \times 3$ em forma de cruz (também semelhante ao desenho da relação adjacência na Figura 2.1(d)) ofereceu os melhores resultados. Gradientes mais "fracos" (como a média dos três gradientes direcionais para imagens 3D) permitem vazamentos indesejados no processo de imersão, e gradientes mais "grossos" (como gradientes morfológicos com elemento estruturante maior) causam perda desnecessária de resolução da imagem.

Outro procedimento de pré-processamento importante em imagens médicas é o cancelamento de objetos indesejados com regiões de alta freqüência, como ossos e vasos sangüíneos em imagens de ressonância magnética (Figura 4.3), que podem gerar picos na imagem de gradiente que tornam a segmentação mais difícil. Este cancelamento pode ser obtido de diversas maneiras, como limiarização, filtro de resposta Gaussiana (*Gaussian Stretching*, a filtragem da imagem com uma curva de resposta como aquela gerada pela Equação 4.3 e mostrada na Figura 4.2) e filtros passa-baixa. A aplicação de cada filtro na segmentação de objetos específicos será discutida no próximo capítulo.

4.3.2 Conexidade Fuzzy Relativa

A principal vantagem da Conexidade Fuzzy é trabalhar diretamente sobre a imagem original, sem a aplicação obrigatória de qualquer filtragem. Em algumas situações o uso de filtros de suavização, como passa-baixa e filtro de medianas pode melhorar a homogeneidade no interior dos objetos, entretanto estes filtros podem acentuar os artefatos de volume parcial (Figura 4.4), "contaminando" voxels vizinhos àqueles que sofrem do efeito. A filtragem para objetos específicos será discutida no próximo capítulo, bem como



(a) (b)

Figura 4.3: Exemplos de regiões com artefatos de alta freqüência e/ou alta intensidade em ressonância magnética: (a) Vasos sangüíneos; (b) Osso e pele.

a seleção dos parâmetros (μ_o, σ_o) pelo usuário. Embora o operador DIFT-CFR evite em muitos casos o passo de pré-processamento, o custo computacional para calcular as Equações 4.4–4.6 torna-o mais lento que o Watershed. Comparações de performance entre os dois operadores serão discutidas em detalhes no próximo capítulo.

(a)

(b)

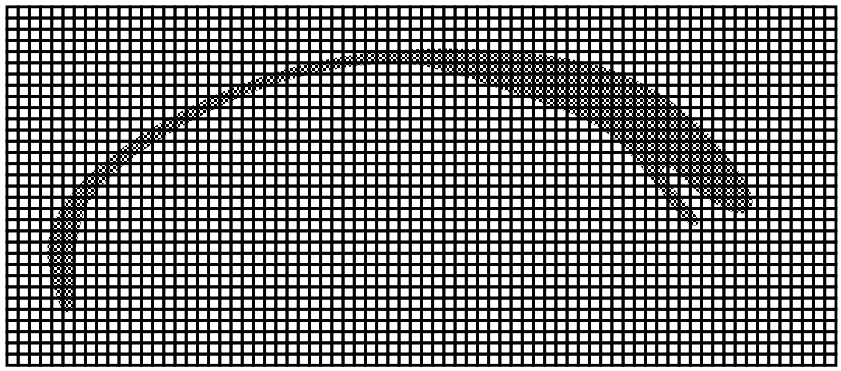


Figura 4.4: Efeito de volume parcial: devido à resolução de aquisição, ocorre *aliasing* na imagem quando dois tecidos distintos ocorrem dentro do mesmo pixel.

Capítulo 5

Aplicação em segmentação de imagens médicas

Os operadores de segmentação implementados através da DIFT, descritos no Capítulo 4, foram avaliados através da segmentação de estruturas anatômicas de interesse neurológico em volumes de ressonância magnética provenientes de exames realizados no Hospital das Clínicas da Unicamp.

Neste capítulo discutimos os procedimentos de pré-processamento adequados para cada combinação de operador e objeto, e apresentamos os resultados dos experimentos realizados ao longo deste trabalho com diversas combinações de objetos de interesse.

5.1 O Software de Segmentação Interativa IVS

Todos os experimentos descritos neste capítulo foram realizados no IVS, software para segmentação interativa de volumes desenvolvido ao longo deste trabalho. O IVS foi implementado em linguagem C, com interface gráfica baseada na biblioteca GTK+. Todos os experimentos foram realizados em PCs com sistema operacional GNU/Linux. Uma versão do IVS está disponível na Internet em `http://www.ic.unicamp.br/~afalcao/ivs` e o seu guia de uso [5] descreve com detalhamento razoável as suas operações. Descreveremos aqui apenas alguns exemplos da interface com que o usuário interage durante a segmentação e um resumo da tarefa de segmentação, do ponto de vista do usuário.

Para segmentar um volume, o usuário deverá:

(1) Carregar o volume a partir de um arquivo em disco.

(2) Aplicar quaisquer filtros de pré-processamento desejados.

(3) Selecionar o operador de segmentação (DIFT/Watershed ou DIFT/CFR).

(4) Alterar a vista 3D para que mostre o corte ou renderização desejado.

(5) Marcar sementes sobre a vista, usando o mouse.

(6) No caso do DIFT-CFR, escolher os parâmetros (μ, σ).

(7) Executar a primeira DIFT.

(8) Alterar a vista 3D para que mostre o corte ou renderização desejado.

(9) Marcar sementes a serem adicionadas ou selecionar árvores a serem removidas.

(10) Executar uma iteração da DIFT

(11) Avaliar o resultado da segmentação, voltando ao passo (8) tantas vezes quanto necessário, até obter uma segmentação aceitável.

Os passos (3)-(11) são realizados na interface ilustrada na Figuras 5.1(a–c). As Figuras ilustram a vista de cortes opacos (usada para a primeira seleção de sementes), a vista de bordas em cortes e a vista de renderização 3D. Os diversos objetos segmentados são diferenciados por cores.

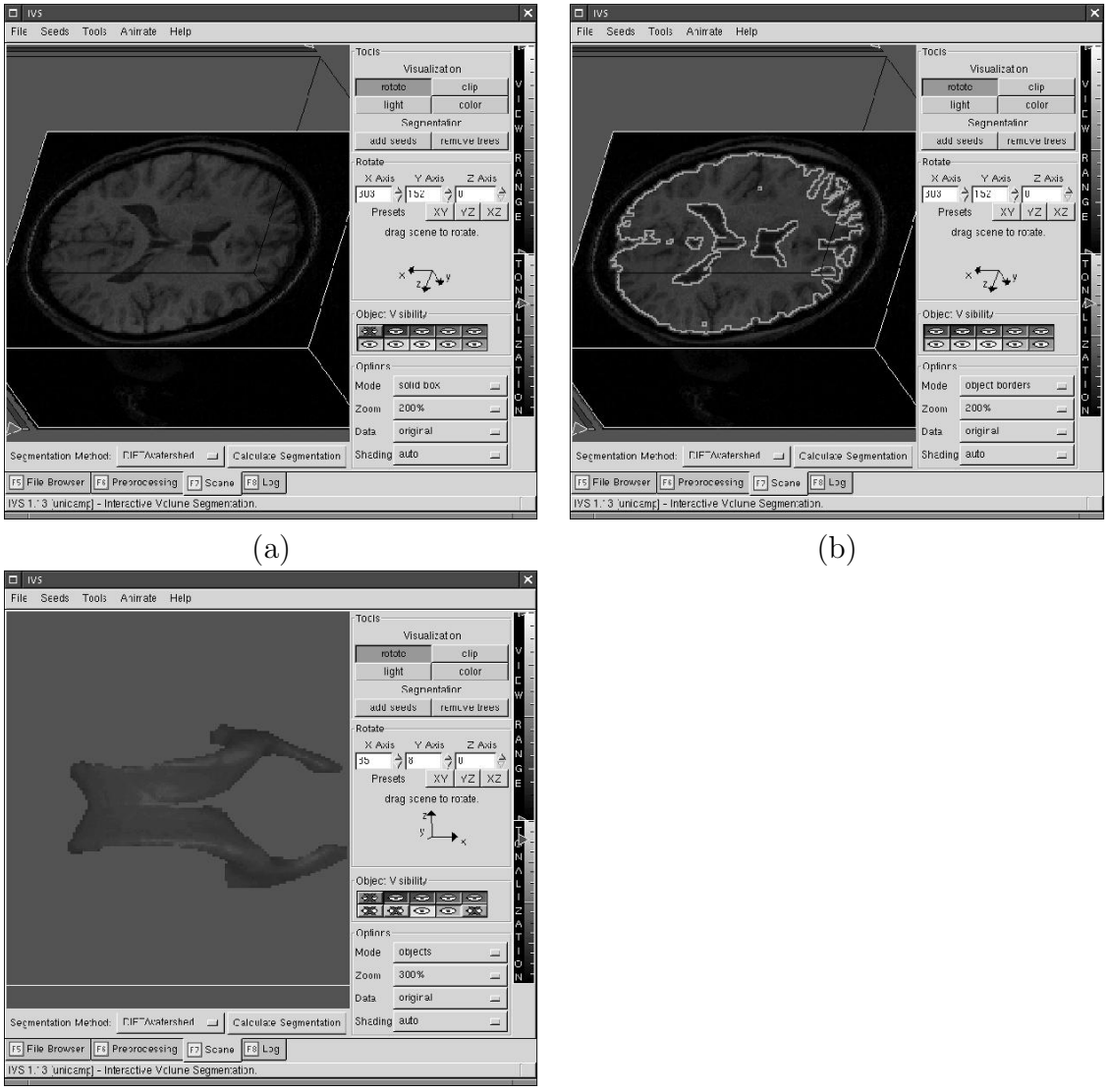


Figura 5.1: Exemplos das vistas disponíveis no IVS durante a segmentação: (a) cortes opacos; (b) bordas em cortes; (c) projeção 3D.

5.2 Pré-processamento para o DIFT-Watershed

Para segmentar imagens em que os objetos são caracterizados por intensidades homogêneas e as bordas entre objetos são caracterizadas por variações bruscas de intensidade, o operador de watershed requer uma imagem de gradiente calculada a partir da cena original. Existem diversos métodos para computar uma imagem de gradientes [3, 25, 12], e durante este trabalho analisamos 4 métodos: gradiente morfológico com elemento estruturante em forma de cruz (semelhante à Figura 2.1), gradiente morfológico com elemento estruturante em forma de cubo $3 \times 3 \times 3$, gradiente direcional máximo e gradiente direcional médio.

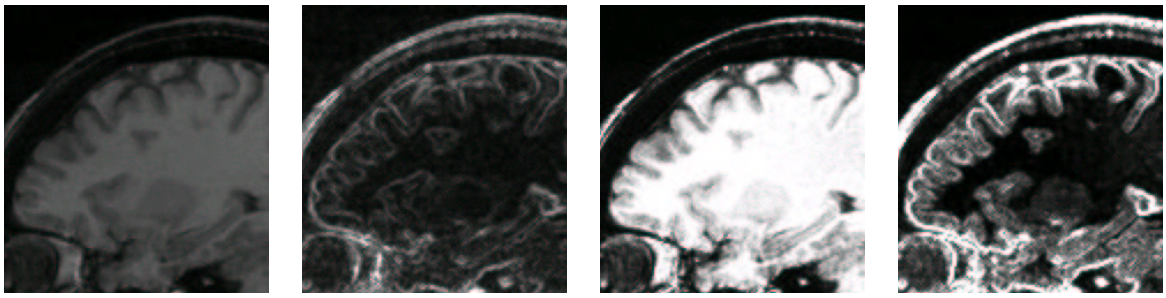
O gradiente morfológico é calculado através da diferença entre a imagem original dilatada por um elemento estruturante e a imagem original erodida pelo mesmo elemento estruturante [25]. O gradiente direcional associa a cada voxel 3 intensidades de gradiente, associadas às variações ao longo dos eixos X, Y e Z do volume. A intensidade do gradiente em X de um voxel $[x, y, z]$ é obtida pela diferença entre as intensidades de $[x + 1, y, z]$ e $[x - 1, y, z]$. A imagem de gradiente direcional médio usa como gradiente de cada voxel a média entre seus gradientes X, Y e Z. A imagem de gradiente direcional máximo usa como gradiente de cada voxel o valor máximo entre seus gradientes X, Y e Z. O gradiente morfológico com elemento estruturante em forma de cruz mostrou-se o mais eficiente: os gradiente direcionais resultaram em bordas quebradas com diversos pontos de "vazamento", e o gradiente morfológico com elemento estruturante em forma de cubo causou perda de resolução em estruturas finas, por gerar bordas muito grossas.

As imagens de gradiente obtidas a partir da imagem original, entretanto, mostraram-se ineficazes em situações em que a borda entre os objetos muda de intensidade gradualmente (devido ao efeito de volume parcial, por exemplo, ou por característica do próprio objeto). Isto ocorre, por exemplo, na borda entre a substância cinzenta (GM) e o fluido cérebro-espinhal (CSF) (Figura 5.2(a)–(b)). Para contornar esta situação, que resulta em intensidades fracas de gradiente nestas bordas, aplicamos um filtro de *stretching* Gaussiano à imagem, antes de calcular a imagem de gradiente. O *stretching* Gaussiano modifica as intensidades de cada voxel p de $I(p)$ para $I'(p)$ de acordo com a Equação 5.1.

$$I'(p) = K \cdot exp\left(\frac{-(I(p) - \mu)^2}{2\sigma^2}\right) \tag{5.1}$$

As Figuras 5.2(c)–(d) ilustram o resultado do filtro na fronteira entre GM e CSF. Em geral desejamos aplicar um stretching Gaussiano com média μ próxima dos valores de intensidade dentro do objeto de interesse e um desvio padrão σ que permita que todos os voxels do objeto tenham alta intensidade ($I'(p) > 0.5$, com $K = 1$).

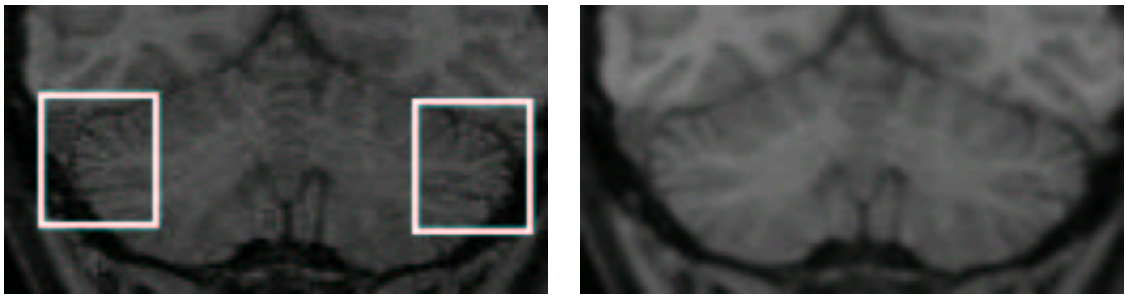
Alguns objetos apresentam bordas texturizadas, devido à presença de estruturas finas demais para a resolução de aquisição. Um exemplo disso são as ramificações na superfície do cerebelo (Figura 5.3). Este tipo de textura, que pode levar a uma imagem de gradiente



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

Figura 5.2: Exemplo de pré-processamento para segmentação do cérebro: (a) Imagem original; (b) Gradiente morfológico calculado sobre a imagem original; (c) Filtro de stretching Gaussiano aplicado sobre a imagem original; (d) Gradiente morfológico calculado a partir de (c).

irregular, pode ser suavizada com um filtro passa-baixa. Neste trabalho usamos uma convolução Gaussiana com máscara $3 \times 3 \times 3$.



(a) (b)

Figura 5.3: Exemplo de atenuação de borda texturizada com filtro passa-baixa: (a) Imagem original. (b) Após aplicação de filtro passa-baixa.

5.3 Resultados obtidos com o DIFT-Watershed

Em todas as descrições de experimentos a seguir, utilizaremos o termo *correção* para denotar as iterações realmente diferenciais da DIFT, isto é, todas as iterações exceto a primeira.

5.3.1 Tempo Linear da DIFT

Nesta primeira série de experimentos segmentamos simultaneamente Cérebro (WM+GM), Cerebelo, Ventrículos Laterais e o Pons-Medula (4 objetos, ver Apêndice B para sua localização) em uma imagem interpolada em diversos tamanhos, para demonstrar a linearidade de tempo da DIFT, bem como a relação entre o tempo requerido para a primeira iteração da DIFT e as seguintes [17]. Os experimentos foram realizados em um PC Pentium 4 de 1.5 GHz com 1280 MB de RAM. A Tabela 5.1 mostra as dimensões das imagens e os tempos obtidos. O volume original é aquele com $181 \times 217 \times 181$ voxels, os outros 5 foram obtidos por interpolação. O pré-processamento usado para esta série de segmentações foi uma sequência de *stretching* Gaussiano (com μ próxima dos valores de intensidade da WM), suavização com uma convolução Gaussiana e o gradiente morfológico com elemento estruturante em cruz. A imagem original foi adquirida em modalidade T1 de ressonância magnética no Hospital das Clínicas da Unicamp, em um paciente sem anomalias conhecidas (controle).

A Figura 5.4 mostra renderizações dos objetos segmentados nos 6 volumes, e a Figura 5.5 mostra um gráfico dos tempos em função do tamanho da cena ($|\mathcal{I}|$).

Dimensões do Volume	Número de Iterações	Tempo da Primeira DIFT	Tempo Médio para Correções	Tempo Total de CPU da DIFT
$121 \times 145 \times 121$	21	3,87s	0,55s	14,80s
$181 \times 217 \times 181$	28	13,67s	1,72s	60,01s
$201 \times 241 \times 201$	23	18,50s	2,30s	69,06s
$241 \times 289 \times 241$	13	34,42s	3,95s	81,79s
$273 \times 328 \times 273$	12	50,41s	5,88s	115,12s
$361 \times 433 \times 361$	12	118,71s	12,32s	254,23s

Tabela 5.1: Resultados da primeira série de experimentos: Verificação de Linearidade do DIFT-Watershed.

Este experimento valida empiricamente nossas afirmações de que a DIFT é computada em tempo linear $O(|\mathcal{I}|)$ e que as iterações diferenciais (correções) levam tempo muito inferior ao tempo requerido pela IFT ou pela primeira iteração da DIFT. (Capítulo 3)

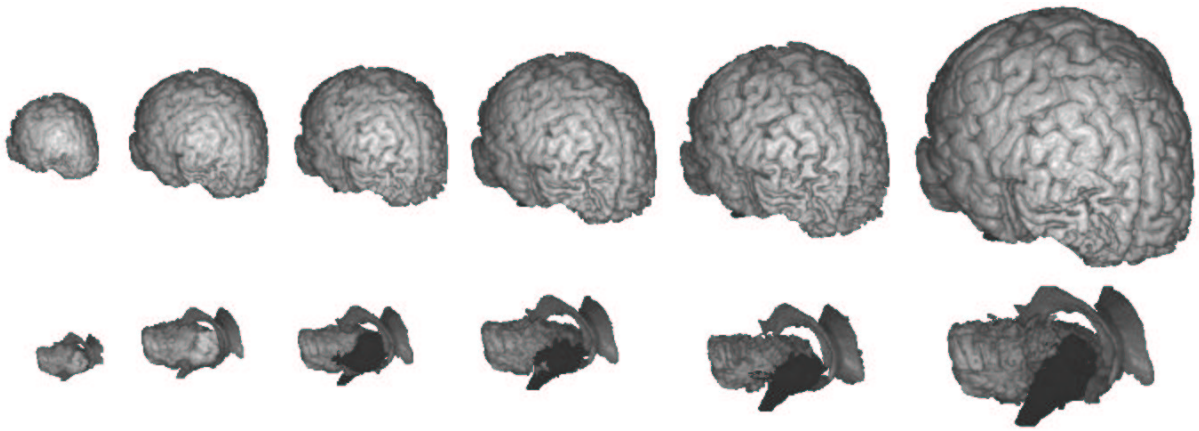


Figura 5.4: Renderizações dos objetos segmentados na primeira série de experimentos.

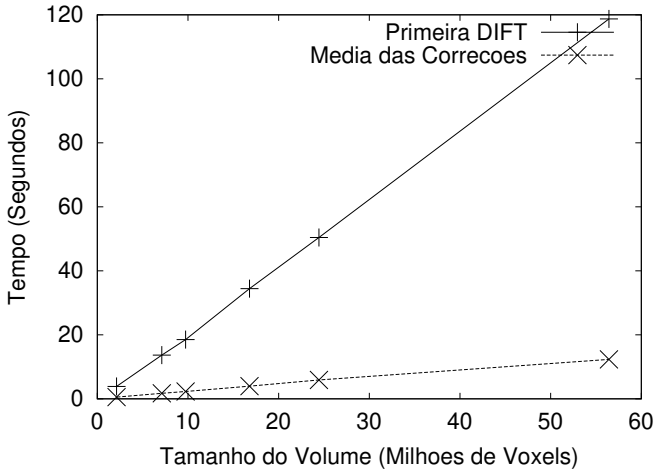


Figura 5.5: Primeira série de experimentos: Tempo de processamento vs. número de voxels.

5.3.2 Segmentação em Pacientes Diversos

Nesta segunda série de experimentos selecionamos 10 volumes, obtidos de 10 pacientes diferentes, sem anomalias conhecidas (controles) e segmentamos os mesmos 4 objetos do experimento anterior (Cérebro, Cerebelo, Ventrículos laterais e Pons-Medula) em todas as cenas [4]. Todos os volumes foram adquiridos em modalidade T1 de ressonância magnética, com um voxels de $0.98 \times 0.98 \times 1.00$ mm, e interpolados para um tamanho de voxel isotrópico de 0.98^3 mm antes de serem segmentados. Foi utilizado o mesmo procedimento de pré-processamento descrito no experimento de linearidade. Esta série de experimentos foi realizada em um PC Pentium 4 de 1.5 GHz com 1280 MB de RAM. Todos os 10 volumes usados nestes experimentos tinham dimensão $181 \times 217 \times 181$ (7.1×10^6 voxels) após a interpolação isotrópica, e intensidades quantizadas em 12 bits (0–4095).

A Tabela 5.2 apresenta os resultados dos experimentos. O Tempo de espera (5a. coluna) é medido somando o tempo da correção (iteração da DIFT) com o tempo de renderização da vista 3D, resultando portanto no tempo de espera do usuário entre o momento em que a correção é requisitada até o momento em que o resultado é apresentado na tela. O Tempo total para segmentação (6a. coluna) é o tempo decorrido desde o início da leitura do volume do disco até a aceitação do resultado como uma segmentação correta, e inclui tempo de leitura do volume do disco, aplicação dos filtros de pré-processamento e todo ciclo de segmentação interativa (seleção de sementes para adição e árvores para remoção, mudança dos parâmetros de renderização, aplicação de iterações da DIFT).

Volume	Número de Iterações	Tempo da Primeira DIFT	Tempo das correções mín-máx (média)	Tempo Médio de Espera	Tempo Total para Segmentação
1	35	18,86s	1,58s–1,84s (1,67s)	2,51s	21m48s
2	40	16,64s	1,59s–1,82s (1,65s)	2,35s	21m28s
3	23	17,42s	1,55s–1,71s (1,60s)	2,48s	15m16s
4	29	17,46s	1,56s–1,81s (1,62s)	2,11s	14m23s
5	32	18,78s	1,57s–1,77s (1,62s)	2,40s	17m03s
6	39	17,85s	1,57s–1,74s (1,61s)	2,15s	17m20s
7	31	18,47s	1,56s–1,95s (1,63s)	2,06s	15m11s
8	35	20,87s	1,56s–2,24s (1,66s)	2,39s	17m41s
9	23	18,86s	1,59s–2,09s (1,69s)	2,13s	11m43s
10	34	18,59s	1,69s–2,49s (1,79s)	2,16s	26m01s

Tabela 5.2: Resultados da segunda série de experimentos: DIFT-Watershed em pacientes diversos.

Um resultado importante deste experimento é a consistência dos tempos de espera

em voluems diferentes. Embora seja possível que uma correção leve tanto tempo quanto a primeira DIFT, observa-se na prática que isto não ocorre. E mesmo quando algumas correções requerem um tempo maior, estas são a minoria, como será visto nas seções seguintes, na apresentação dos experimentos com o operador DIFT-Fuzzy, em que correções mais longas ocorreram.

Outro resultado importante é o tempo médio de espera, mantido abaixo dos 3 segundos, mostrando que é viável realizar segmentação 3D interativa em um PC de baixo custo, sem hardware especializado (nem mesmo aceleração gráfica 3D por hardware é usada para realizar a renderização 3D, feita inteiramente por software, com os algoritmos descritos no Capítulo 6). Ainda que o tempo total de segmentação mantenha-se em torno de 20 minutos por volume, este tempo é gasto em um processo interativo, em que o usuário pode garantir que obterá uma segmentação adequada ao final do procedimento através das decisões tomadas a cada correção. O usuário pode, inclusive, terminar a segmentação mais cedo, com uma segmentação imperfeita, que pode ser suficiente para alguns propósitos clínicos. Não é necessário segmentar perfeitamente cada sulco para obter uma estimativa do volume do cérebro, por exemplo.

A realização das mesmas segmentações com o Watershed implementado através da IFT requereria aproximadamente 8 vezes mais tempo de CPU, além de inviabilizar (ou pelo menos tornar muito tediosa) a segmentação interativa, com tempos de espera em torno de 19 segundos. A Tabela 5.3 mostra o tempo de CPU que seria requerido para realizar as mesmas segmentações com a IFT em vez da DIFT.

Volume	Tempo de CPU com DIFT	Tempo de CPU com IFT	Ganho de Eficiência
1	75,54s	660,10s	8,74
2	80,90s	665,60s	8,23
3	52,68s	400,66s	7,61
4	62,88s	506,34s	8,05
5	69,02s	600,96s	8,71
6	79,06s	696,15s	8,81
7	67,43s	572,57s	8,49
8	77,41s	730,45s	9,44
9	56,00s	433,78s	7,75
10	77,55s	632,06s	8,15

Tabela 5.3: Ganho de eficiência da DIFT sobre a IFT, nas segmentações da segunda série de experimentos.

A Figura 5.6 mostra renderizações 3D dos objetos segmentados em 5 dos 10 volumes,

e a Figura 5.7 mostra as bordas entre objetos em alguns cortes, ilustrando a qualidade das segmentações obtidas.

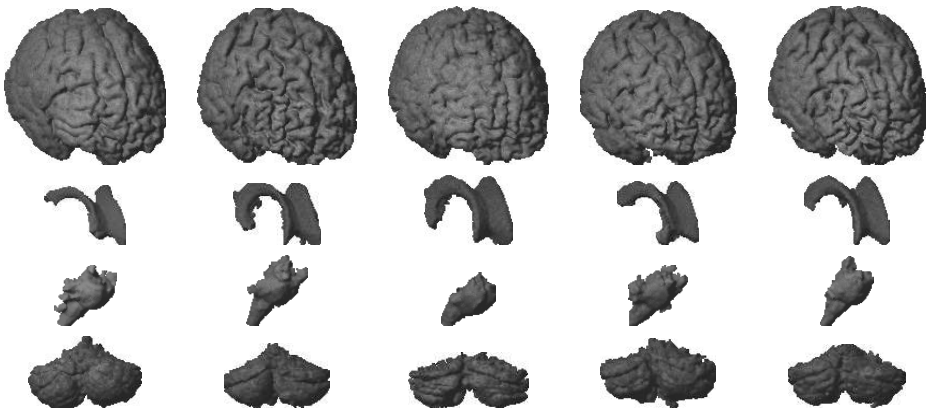


Figura 5.6: Renderizações 3D dos objetos segmentados em 5 dos 10 volumes na segunda série de experimentos. De cima para baixo: cérebro, ventrículos laterais, pons-medula e cerebelo; Da esquerda para a direita: volumes 1, 3, 5, 7 e 9.

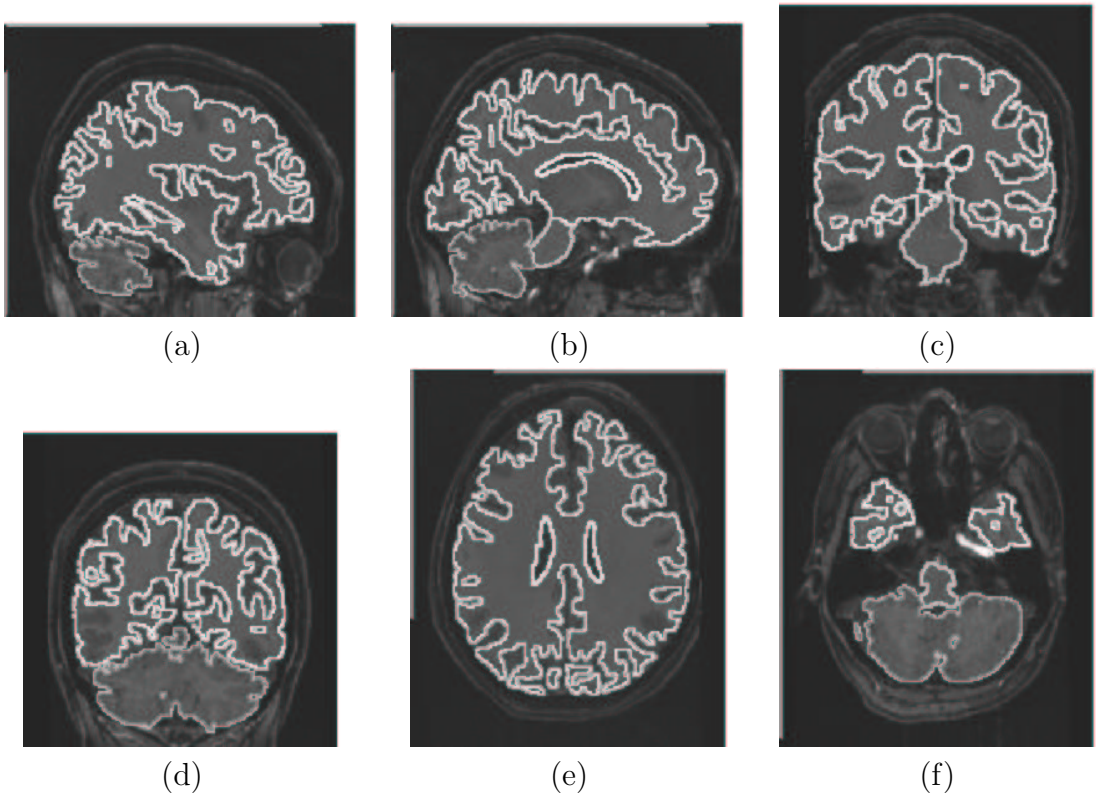


Figura 5.7: Exemplo da qualidade das segmentações obtidas na segunda série de experimentos: bordas dos objetos em cortes ortogonais.

5.4 Pré-processamento para o DIFT-CFR

A conexidade fuzzy relativa (CFR) opera diretamente sobre a cena de intensidades, sem a necessidade de cálculo de uma imagem de gradiente. A eficiência da CFR para a segmentação depende essencialmente de quão homogêneas são as intensidades dentro de cada objeto. Filtros de suavização e de redução de ruído podem ajudar a homogenizar o interior dos objetos de interesse.

Entretanto, apesar de poder ser aplicado sobre a imagem original, o DIFT-CFR exige que o usuário selecione k pares (μ_i, σ_i) antes de realizar a primeira iteração de segmentação. No IVS, a seleção destes parâmetros é realizada na caixa de diálogo ilustrada na Figura 5.8. As fronteiras entre as zonas de influência de cada par (μ_i, σ_i) são mostradas instantaneamente na área de visualização, orientando o usuário para a seleção de parâmetros adequados. Na prática, os melhores resultados de segmentação para k objetos foram obtidos com $k + 2$ pares (μ_i, σ_i), k pares representando os objetos propriamente

ditos, um par representando regiões de fundo mais escuras que os objetos e outro par representando regiões de fundo mais claras que os objetos.

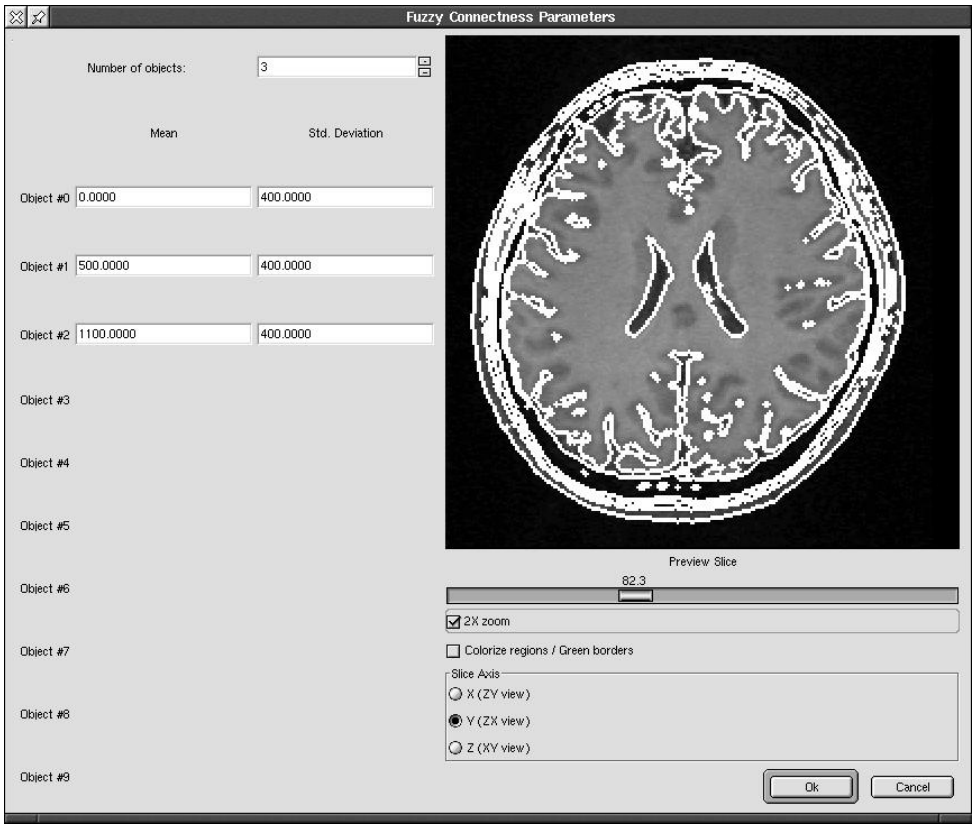


Figura 5.8: Caixa de diálogo do IVS usada para a seleção dos parâmetros do DIFT-CFR. Neste exemplo, uma seleção adequada para segmentar os ventrículos laterais.

5.5 Resultados obtidos com DIFT-CFR

Nesta terceira série de experimentos, selecionamos 10 volumes de controles (pacientes sem anomalias conhecidas), adquiridas em modalidade T1, tamanho de voxel $0.98 \times 0.98 \times 1.00$ mm e quantizadas em 12 bits, interpolamo-nos para um tamanho de voxel isotrópico de 0.98^3 mm e, antes da segmentação, cortamos cada volume para excluir áreas sem objetos (regiões vazias acima e dos lados da cabeça, e região abaixo do cerebelo). As dimensões dos volumes resultantes, usados para as segmentações, são apresentadas na Tabela 5.4.

Segmentamos em cada volume, separadamente, os ventrículos laterais e o núcleo caudado. Esta série de experimentos foi realizada em um PC Athlon (Thunderbird) de

Volume	Dimensões	Número de Voxels
1	$253 \times 161 \times 161$	6 558 013
2	$225 \times 163 \times 166$	6 088 050
3	$228 \times 175 \times 159$	6 344 100
4	$225 \times 175 \times 173$	6 811 875
5	$230 \times 162 \times 159$	5 924 340
6	$230 \times 168 \times 165$	6 375 600
7	$229 \times 181 \times 217$	8 994 433
8	$234 \times 183 \times 195$	8 350 290
9	$216 \times 161 \times 152$	5 285 952
10	$213 \times 164 \times 160$	5 589 120

Tabela 5.4: Dimensões dos volumes usados na terceira série de experimentos.

1100 MHz com 384 MB de RAM. Para a segmentação dos ventrículos não foi usado qualquer pré-processamento. Para a segmentação do núcleo caudado foi usado um filtro de mediana com máscara $3 \times 3 \times 3$.

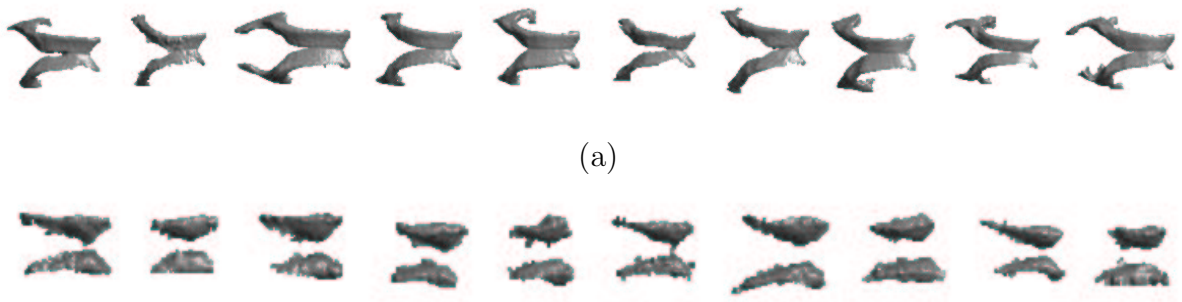
Os resultados obtidos são mostrados nas Tabelas 5.5 e 5.6. Embora estas segmentações tenham sido realizadas em um computador mais lento que o dos experimentos anteriores, o DIFT-CFR é mais lento que o DIFT-Watershed, devido à necessidade de avaliar a expressão da Equação 4.6 no DIFT-CFR. Considerando apenas a primeira DIFT, este Athlon de 1100 MHz processa o volume a uma taxa de 196 742 voxels/segundo com o DIFT-CFR, e a uma taxa de 350 075 voxels/segundo com o DIFT-Watershed. Isto faz o DIFT-Watershed 77% mais rápido que o DIFT-CFR. Durante os experimentos com o DIFT-CFR, algumas sementes colocadas dentro dos objetos conquistaram boa parte do fundo, causando correções mais longas. Mesmo assim, de um total de 673 correções realizadas nas 20 segmentações, apenas 16 causaram um tempo de espera maior que 5 segundos. Com 97% das correcões requerendo menos que 5 segundos de espera, podemos concluir que a interatividade da tarefa de segmentação não é comprometida. A Figura 5.9 mostra renderizações dos ventrículos (a) e núcleos caudados segmentados (b). A Figura 5.10 mostra exemplos das bordas obtidas, em cortes ortogonais.

Volume	Número de Iterações	Tempo da Primeira DIFT	Tempo Médio para Correções	Tempo Médio de Espera	Tempo Total para Segmentação
1	23	29,91s	1,73s	2,76s	12m56s
2	48	30,19s	1,77s	2,73s	15m20s
3	19	31,22s	2,86s	3,96s	8m29s
4	14	34,59s	1,65s	2,62s	8m41s
5	18	30,73s	1,58s	2,60s	8m29s
6	38	29,57s	1,76s	2,79s	13m35s
7	14	41,42s	2,35s	3,81s	7m41s
8	15	44,65s	2,10s	3,34s	11m44s
9	24	25,63s	1,50s	2,42s	11m07s
10	21	28,54s	1,47s	2,37s	10m49s

Tabela 5.5: Resultados das segmentações de ventrículos laterais com DIFT-CFR na terceira série de experimentos.

Volume	Número de Iterações	Tempo da Primeira DIFT	Tempo Médio para Correções	Tempo Médio de Espera	Tempo Total para Segmentação
1	62	34,06s	1,73s	2,76s	19m51s
2	44	31,62s	1,72s	2,62s	15m39s
3	50	34,25s	1,82s	2,74s	21m07s
4	47	36,46s	2,42s	3,53s	17m45s
5	44	30,01s	2,86s	3,70s	16m19s
6	56	31,62s	2,70s	3,78s	19m36s
7	26	48,34s	3,95s	5,27s	14m35s
8	35	44,52s	3,36s	4,69s	14m30s
9	24	27,59s	1,40s	2,32s	9m44s
10	71	29,34s	2,23s	3,16s	21m49s

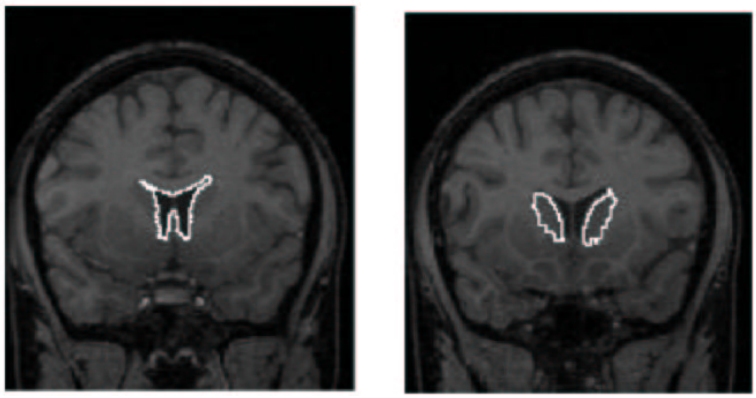
Tabela 5.6: Resultados das segmentações de núcleo caudado com DIFT-CFR na terceira série de experimentos.



(a)

(b)

Figura 5.9: Renderizações dos ventrículos laterais (a) e núcleos caudados (b) segmentados com DIFT-CFR na terceira série de experimentos.



(a) (b)

Figura 5.10: Exemplos das bordas entre objeto e fundo resultantes das segmentações da terceira série de experimentos: (a) ventrículos laterais; (b) núcleo caudado.

5.6 Segmentação simultânea de Múltiplos Objetos

Ainda que os operadores baseados na DIFT ofereçam um bom tempo de resposta às correções interativas, o tempo total requerido para completar a segmentação pode ser considerado alto em alguns casos, chegando a quase 22 minutos para segmentar um único objeto, no caso do núcleo caudado na terceira série de experimentos. Uma forma eficaz de reduzir o tempo requerido para segmentar cada objeto é segmentar vários objetos simultanemente, conforme foi feito nas duas primeiras séries de experimentos. Desta forma, o tempo de verificação da segmentação é "compartilhado" por todos os objetos. Com o DIFT-Watershed, isto requer um procedimento de pré-processamento que gere boas bordas para todos os objetos a segmentar, o que pode ser uma tarefa complexa para alguns conjuntos de objetos. Com o DIFT-CFR, a segmentação de múltiplos objetos simultaneamente é prevista pelo próprio modelo de conexidade fuzzy, que admite diversos pares (μ, σ), representando diversos objetos.

Repetimos as segmentações do núcleo caudado e dos ventrículos laterais nas mesmas 10 cenas, mas desta vez segmentando ambos os objetos ao mesmo tempo com o DIFT-CFR, sem qualquer pré-processamento. Obtivemos sessões de segmentação que duraram em média $21m24s$, resultando em $10m42s$ por objeto. O tempo total de segmentação mostrou-se 30% menor que a soma dos tempos requeridos para segmentá-los separadamente. O ganho de tempo será maior à medida que mais objetos forem segmentados simultaneamente. Em geral, o tempo total da segmentação será limitado inferiormente pelo objeto mais difícil de segmentar (isto é, que requer mais correções).

5.7 Comentários Finais

É importante notar que a maior parte do tempo requerido para completar a segmentação é gasto pelo usuário, verificando a qualidade da segmentação em cortes ortogonais do volume, mudando os parâmetros de visualização e marcando sementes. O tempo do usuário representa de 68% a 75% do tempo total de segmentação. Durante o desenvolvimento da aplicação de segmentação interativa, não nos preocupamos excessivamente com o design da interface com o usuário, pois este não era o foco do trabalho, mas acreditamos que o tempo de segmentação possa ser reduzido consideravelmente com uma interface projetada de forma mais criteriosa.

Neste capítulo, mostramos exemplos práticos de aplicação dos operadores descritos no Capítulo 4, com resultados que mostram que a DIFT claramente viabiliza a segmentação interativa de volumes em computadores de baixo custo, com a vantagem adicional de utilizar operadores clássicos de segmentação, como a Transformada de Watershed, desde que possam ser formulados como uma instância da IFT.

Capítulo 6

Renderização rápida de volumes dinâmicos

A tarefa de segmentação interativa requer a apresentação do estado da segmentação entre cada correção interativa, para que o usuário possa avaliar a qualidade e corretude da segmentação e decidir sua próxima ação. Portanto, a rapidez na renderização da cena após cada correção torna-se tão importante quanto a própria correção.

A literatura apresenta diversos métodos de renderização de volumes [31, 34, 44], porém todos os métodos rápidos requerem a construção de alguma estrutura que organize e indexe a informação do volume de acordo com a segmentação corrente, construção esta que pode ser lenta e exigir um espaço de armazenamento considerável. O uso de métodos de visualização baseados em estruturas *Shell* para renderizar a cena durante a segmentação interativa exigiria a reconstrução do shell a cada iteração.

Neste capítulo apresentamos e discutimos o método de renderização implementado na ferramenta de segmentação interativa descrita no Capítulo 5, que não depende da manutenção (entre renderizações) de qualquer estrutura dependente da segmentação, e independe de recursos especializados de hardware.

6.1 Modelo de Visualização

A inspeção de uma segmentação requer a visualização do volume segmentado a partir de ângulos diversos, bem como a inspeção de planos de corte. É desejável combinar a informação da imagem original (intensidade do sinal) com alguma delineação da segmentação (uso de cores para diferenciar objetos, ou o traçado de bordas nas fronteiras entre objetos).

O método de renderização apresentado neste capítulo produz cenas de projeção ortogonal, em que o volume pode ser rotacionado arbitrariamente para simular a visualização

a partir de qualquer posição ao seu redor. Uma função importante da cena gerada é permitir a seleção de voxels do volume (para selecionar novas sementes ou árvores a serem removidas pela DIFT); Para evitar confusão, é desejável que cada pixel da projeção 2D corresponda a apenas um voxel do volume, para que um clique sobre a imagem selecione, sem ambigüidade, apenas um voxel do volume. Não nos preocupamos, portanto, em permitir transparência parcial: cada voxel pode ser apenas totalmente visível (opaco), ou invisível. A visibilidade de cada voxel é condicionada pelo objeto a que pertence ou por sua pertinência a uma região de visibilidade. A forma mais simples de definir uma região de visibilidade é restringir intervalos de visibilidade sobre os eixos de coordenadas do volume, formando uma caixa de corte; Desta forma, podemos obter a inspeção de planos ortogonais ao sistema de coordenadas do volume.

A percepção tridimensional do volume requer a aplicação de um modelo de iluminação à cena. Para obter algum realismo, é necessário estimar, em cada voxel projetado, um vetor normal à superfície naquele local, para determinar quanta luz é refletida na direção do observador. Neste método, os vetores normais são estimados no espaço bidimensional da projeção, solução mais eficiente que realizar a estimativa no espaço tridimensional ou reduzir a superfície projetada a polígonos.

6.2 Estágios de Renderização

A Figura 6.1 mostra os 4 estágios necessários para completar a renderização do volume em uma imagem bidimensional.

O objetivo da renderização é gerar uma vista 2D a partir da cena 3D e de um conjunto de parâmetros. A vista 2D inclui uma imagem 2D e um mapa que relacione cada pixel 2D ao voxel que ele representa (ou a um marcador *nil*, indicando que não há voxel projetado naquela posição da imagem).

A Figura 6.2 mostra as relações entre as coordenadas da cena 3D e da vista 2D. A projeção ortogonal é obtida com a aplicacação de uma transformação T do volume (em geral, uma composição de rotações) e com o descarte da coordenada Z. Assim temos um observador fixo em $\langle 0, 0, -\infty \rangle$.

É importante notar a diferença de tamanho entre o espaço da cena ($W_{VOL} \times H_{VOL} \times D_{VOL}$ voxels) e o espaço da vista ($W_{VISTA} \times H_{VISTA}$). Varrer o espaço da cena leva muito mais tempo que varrer o espaço da vista, e a principal estratégia deste método para acelerar a renderização é extrair toda informação necessária da cena 3D em apenas uma varredura e armazená-la em uma estrutura 2D, permitindo que o restante da renderização trabalhe sobre o espaço da vista. Esta varredura é realizada no primeiro estágio (Projeção) e a estrutura 2D resultante é o buffer de cena 2D.

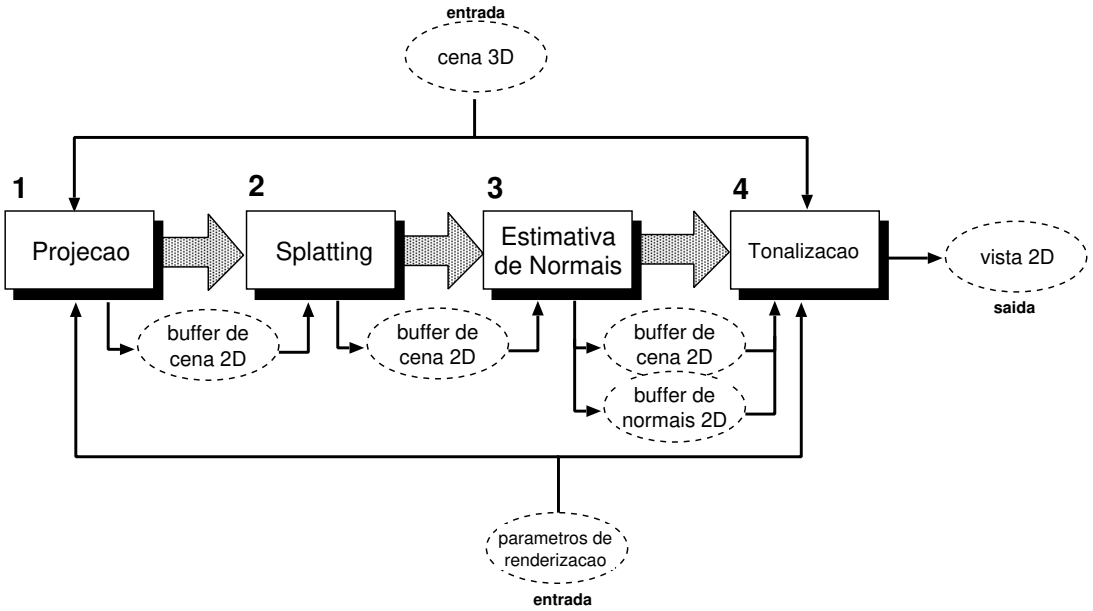


Figura 6.1: Estágios de Renderização

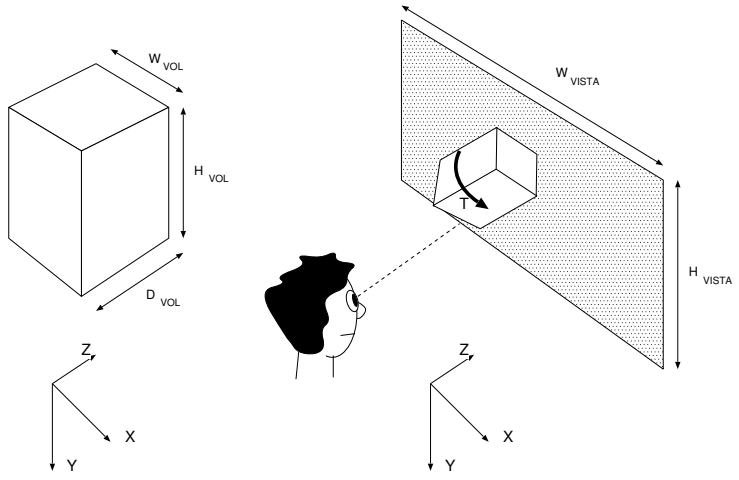


Figura 6.2: Sistemas de coordenadas

6.3 Projeção

Este estágio tem por objetivo varrer a cena 3D e criar o buffer de cena 2D com uma projeção inicial dos voxels visíveis. As condições para visibilidade de um voxel são dadas nos parâmetros de renderização. Os dois tipos de condição usados na ferramenta de segmentação interativa desenvolvida neste trabalho são:

Visibilidade de cada rótulo. Os objetos segmentados são diferenciados por um mapa de rótulos. O usuário pode selecionar quais objetos devem ser projetados e quais devem ser considerados invisíveis.

Caixa de corte. São fornecidos intervalos de coordenadas X, Y e Z que definem uma caixa de visibilidade. Voxels fora da caixa são considerados invisíveis.

O buffer de cena 2D, B, armazena, para cada pixel (x, y) do espaço de coordenadas da vista: voxel ali projetado ($\langle B_{(x,y)}.X, B_{(x,y)}.Y, B_{(x,y)}.Z\rangle$), seu rótulo ($B_{(x,y)}.L$) e suas coordenadas no espaço transformado pela transformação T ($\langle B_{(x,y)}.X_T, B_{(x,y)}.Y_T, B_{(x,y)}.Z_T\rangle$). Como há um mapeamento um para um entre voxels da cena e pixels da vista, a transformação T não deve aumentar ou diminuir as dimensões dos voxels, sendo restrita a uma composição arbitrária de rotações e translações. Transformações de escala para obter ampliação (zoom) da vista, se desejadas, são realizadas no último estágio de renderização (Pós-processamento).

O estágio de projeção varre a cena 3D, varrendo os eixos X, Y e Z de forma a visitar primeiro os voxels mais próximos do observador (front-to-back). O algoritmo deste estágio é apresentado abaixo, de forma simplificada:

Algoritmo 6-1 – PROJEÇÃO

ENTRADA: Volume $\mathcal{I}$, Trasformação T, condições de visibilidade
SAÍDA: Buffer de cena 2D B

1. *Inicialize B com nil em todos os pixels.*
2. *Determine a direção de varredura front-to-back para os eixos X, Y e Z.*
3. **Para Cada** $\langle X, Y, Z\rangle \in \mathcal{I}$, *varridos front-to-back* **Faça**
4. ⎸ **Se** $\langle X, Y, Z\rangle$ *for visível* **Então**
5. ⎸ ⎸ *Calcule* $P \leftarrow \langle X, Y, Z\rangle \times T$
6. ⎸ ⎸ **Se** $P.Z < B_{(P.X,P.Y)}.Z_T$ **Então**
7. ⎸ ⎸ ⎸ *Atualize* $B_{(P.X,P.Y)}$ *com* $\langle X, Y, Z\rangle$ *e* P.

A determinação da direção de varredura (linha 2) é trivial: aplica-se a transformada T a dois pontos extremos ao longo do eixo, por exemplo $p = \langle 0, 0, 0\rangle \times T$ e $q = \langle W_{VOL} -$

$1, 0, 0\rangle \times T$ para o eixo X. Se $p.z < q.z$, o eixo X deve ser percorrido em ordem crescente; Caso contrário, deve ser percorrido em ordem decrescente. A varredura front-to-back minimiza os sucessos no teste da linha 6, evitando a execução inútil da linha 7 para voxels que seriam encobertos por outros mais próximos do observador.

Os pixels do buffer B em que nenhum voxel foi projetado são marcados por valor especial de voxel, *nil*.

6.4 Splatting

Quando uma transformação de rotação é aplicada na ordem direta, como no primeiro estágio, é comum a ocorrência de buracos na projeção resultante, devido ao casamento imperfeito dos pixels entre os dois sistemas de coordenadas. Um exemplo deste efeito é mostrado em uma rotação 2D na Figura 6.3. Os pixels da imagem destino que não recebem nenhuma origem de pixel (círculos pretos) não são preenchidos na aplicação direta da transformação.

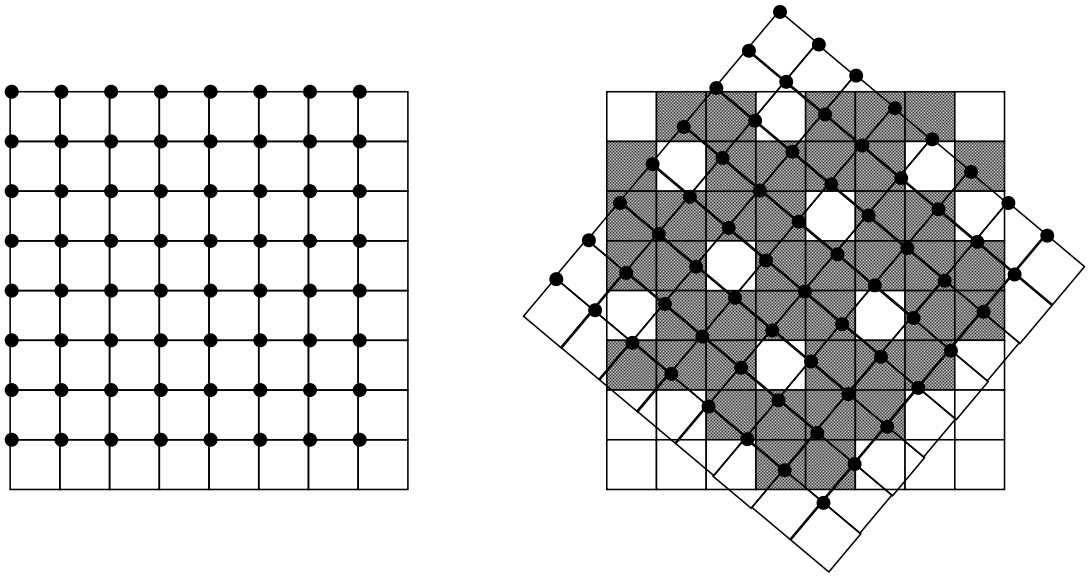


Figura 6.3: Exemplo de formação de buracos na aplicação direta de uma transformada de rotação.

Este segundo estágio tem por objetivo fechar eventuais buracos. O splatting percorre o buffer de cena B da esquerda para a direita e de cima para baixo com uma máscara 2×2. Se o pixel na posição superior esquerda da máscara tiver um voxel projetado, este

voxel projetado se sobrepõe a outros pixels dentro da máscara que possuam Z_T maior que o seu (mais distantes do observador) ou que não tenham nenhum voxel projetado. O algoritmo usado é apresentado abaixo:

Algoritmo 6-2 – SPLATTING

ENTRADA: Buffer de cena 2D B.
SAÍDA: B.

1. **Para** y *de* 0 *a* $H_{VISTA} - 2$ **Faça**
2. **Para** x *de* 0 *a* $W_{VISTA} - 2$ **Faça**
3. **Se** $B_{(x,y)} \neq nil$ **Então**
4. **Se** $B_{(x+1,y)} = nil$ *ou* $B_{(x+1,y)}.Z_T > B_{(x,y)}.Z_T$ **Então**
5. $B_{(x+1,y)} \leftarrow B_{(x,y)}$.
6. **Se** $B_{(x,y+1)} = nil$ *ou* $B_{(x,y+1)}.Z_T > B_{(x,y)}.Z_T$ **Então**
7. $B_{(x,y+1)} \leftarrow B_{(x,y)}$.
8. **Se** $B_{(x+1,y+1)} = nil$ *ou* $B_{(x+1,y+1)}.Z_T > B_{(x,y)}.Z_T$ **Então**
9. $B_{(x+1,y+1)} \leftarrow B_{(x,y)}$.

6.5 Estimativa de Normais

O vetor normal à superfície em cada ponto do buffer de cena pode ser estimado a partir dos triângulos de pixels adjacentes em que um dos vértices seja o pixel centrado no ponto de estimativa (formando um "guarda-chuva" centrado no ponto onde desejamos estimar a normal). Uma estimativa com 8 vizinhos pode ser obtida através da média das normais dos 8 triângulos da Figura 6.4.

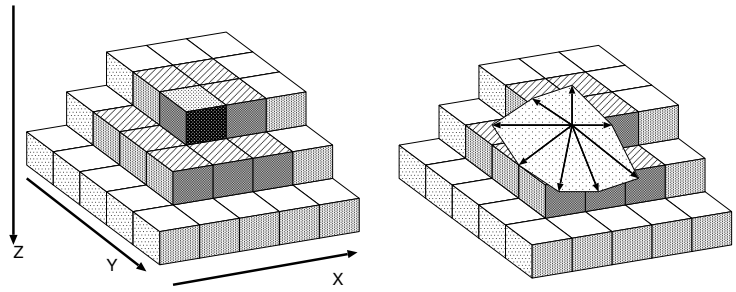


Figura 6.4: Estimativa de normal com 8 vizinhos e 8 triângulos sobre o buffer de cena.

Para obter uma estimativa mais suave, podemos adicionar mais 8 vizinhos e 8 triângulos, conforme ilustrado na Figura 6.5.

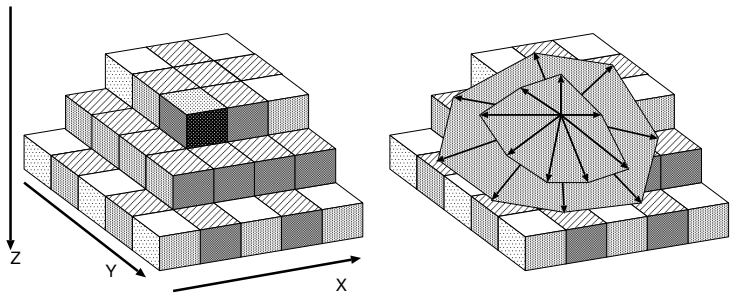


Figura 6.5: Estimativa de normal com 16 vizinhos e 16 triângulos sobre o buffer de cena.

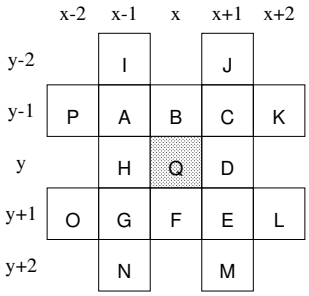


Figura 6.6: Enumeração dos 16 vizinhos usados para a estimativa da normal no buffer de cena.

Os vetores são calculados a partir das coordenadas transformadas $\langle X_T, Y_T, Z_T \rangle$ presentes no buffer de cena. Considerando a nomenclatura da Figura 6.6, o vetor normal $\vec{N}_{(x,y)}$ no voxel projetado no pixel $Q = (x, y)$ é dado pela Equação 6.1.

$$\begin{aligned}
\vec{N}_{(x,y)} \;=\; & \frac{\vec{QB} \times \vec{QA}}{|\vec{QB} \times \vec{QA}|} + \frac{\vec{QC} \times \vec{QB}}{|\vec{QC} \times \vec{QB}|} + \frac{\vec{QD} \times \vec{QC}}{|\vec{QD} \times \vec{QC}|} + \frac{\vec{QE} \times \vec{QD}}{|\vec{QE} \times \vec{QD}|} + \\
& + \frac{\vec{QF} \times \vec{QE}}{|\vec{QF} \times \vec{QE}|} + \frac{\vec{QG} \times \vec{QF}}{|\vec{QG} \times \vec{QF}|} + \frac{\vec{QH} \times \vec{QG}}{|\vec{QH} \times \vec{QG}|} + \frac{\vec{QA} \times \vec{QH}}{|\vec{QA} \times \vec{QH}|} + \\
& + \frac{\vec{QJ} \times \vec{QI}}{|\vec{QJ} \times \vec{QI}|} + \frac{\vec{QK} \times \vec{QJ}}{|\vec{QK} \times \vec{QJ}|} + \frac{\vec{QL} \times \vec{QK}}{|\vec{QL} \times \vec{QK}|} + \frac{\vec{QM} \times \vec{QL}}{|\vec{QM} \times \vec{QL}|} + \\
& + \frac{\vec{QN} \times \vec{QM}}{|\vec{QN} \times \vec{QM}|} + \frac{\vec{QO} \times \vec{QN}}{|\vec{QO} \times \vec{QN}|} + \frac{\vec{QP} \times \vec{QO}}{|\vec{QP} \times \vec{QO}|} + \frac{\vec{QI} \times \vec{QP}}{|\vec{QI} \times \vec{QP}|}
\end{aligned} \tag{6.1}$$

Todos os termos são normalizados para que o resultado reflita uma média das direções das normais dos 16 triângulos.

Conforme será apresentado na próxima seção, o modelo de iluminação usado, baseado no modelo de Phong [24], precisa da normal apenas para obter o cosseno do ângulo entre a normal e a fonte de luz, e o cosseno do dobro deste ângulo. Assim, em vez de armazenar um vetor (3 componentes) para cada pixel do buffer de cena, neste estágio já calculamos o cosseno do ângulo entre cada vetor normal $\vec{N}_{(x,y)}$ e a direção de incidência da fonte de luz.

Este método de renderização assume que a cena é iluminada por uma única fonte de luz pontual no infinito, com a direção de incidência representada por um vetor $\vec{l}$, fornecido nos parâmetros de inicialização.

O cosseno do ângulo entre $\vec{N}_{(x,y)}$ e $\vec{l}$ é obtido através do produto interno:

$$\theta_{(x,y)} = \vec{l} \measuredangle \vec{N}_{(x,y)} \tag{6.2}$$

$$cos\,\theta_{(x,y)} = \left(\frac{\vec{l}}{|\vec{l}|}\right) \cdot \left(\frac{\vec{N}_{(x,y)}}{|\vec{N}_{(x,y)}|}\right) \tag{6.3}$$

Uma matriz de cossenos, de mesma dimensão que o buffer de cena, é calculada e passada para o estágio de Tonalização.

6.6 Tonalização

Este estágio compõe, finalmente, a imagem colorida da vista. Os parâmetros de renderização controlam como cada pixel será tonalizado de acordo com seu rótulo e intensidade do voxel associado na cena 3D, e que modelo de iluminação será aplicado.

6.6.1 Coloração

O usuário pode escolher misturar, em cada pixel da vista, a intensidade do voxel associado na cena 3D com a cor associada ao seu rótulo, em alguma proporção. O resultado é uma imagem em que cada objeto tem uma cor diferente, porém com alguma informação associada à intensidade da imagem original. Ou, então, exibir a intensidade original da cena no interior dos objetos e pintar os pixels de fronteira com as cores associadas aos rótulos, formando bordas de fácil percepção. A decisão borda/não-borda pode ser tomada comparando os rótulos dos 8 vizinhos de cada pixel do buffer de cena; A presença de rótulos diferentes indica que o pixel é uma fronteira entre objetos. Uma variação desta idéia é comparar não rótulos, mas raízes, permitindo o traçado das bordas entre árvores da floresta da (D)IFT.

O primeiro passo da Tonalização, portanto, é compor uma imagem colorida (RGB, 8 bits por componente) de mesma dimensão que o buffer de cena, preenchê-la com uma cor

de fundo (dada pelo usuário, nos parâmetros de renderizaçào) e percorrer o buffer de cena colorizando os pixels da imagem de acordo o modo de coloração pedido nos parâmetros de renderização. A mistura de cores pode ser obtida com médias ponderadas dos valores de componentes R,G e B das cores a serem misturadas.

6.6.2 Iluminação

O próximo passo é escurecer ou clarear os pixels da imagem gerada de acordo com o modelo de iluminação solicitado (exceto pixels de fundo, onde nenhum voxel está projetado). Para a geração de imagens realistas que ofereçam boa percepção espacial é usado um modelo de iluminação semelhante ao modelo de Phong [24], que leva em consideração uma iluminação de ambiente mínima, reflexão difusa, especular e atenuação de luminância para objetos mais distantes do observador. Neste modelo, a cor de cada pixel da imagem composta na coloração é convertida para o modelo YCbCr e a componente de luminância Y é modificada para Y' de acordo com a Equação 6.4.

$$Y'_{(x,y)} = K_a + (\alpha + \beta \cdot \Delta Z^2_{(x,y)}) \cdot Y_{(x,y)} \cdot \left\{ K_d \cdot cos\,\theta_{(x,y)} + K_s \cdot cos^m \left[max\left(\frac{\pi}{2}, 2\theta_{(x,y)}\right)\right]\right\} \quad (6.4)$$

K_a, K_d e K_s são coeficientes de luz ambiente, reflexão difusa e reflexão especular, respectivamente. m é uma característica da curva de reflexão especular do material modelado. α e β modelam a atenuação da luminância em relação à distância entre o objeto e o observador. Todos esses parâmetros são fornecidos pelo usuário como parte dos parâmetros de renderização. $2\theta_{(x,y)}$ é calculado a partir dos valores de $cos\,\theta_{(x,y)}$ obtidos no estágio de estimativa de normais. $\Delta Z_{(x,y)}$ é um coeficiente de profundidade calculado de tal forma que seja 0 para pontos na posição mais distante possível do observador e 1 para pontos na posição mais próxima possível do observador. A maior distância possível entre dois pontos na cena 3D é a diagonal do paralelepípedo do volume. Para um volume centrado na origem, sem transformações de escala, a maior profundidade absoluta será meia diagonal (pois o ponto médio da diagonal do paralelepípedo está na origem); Nesta situação, $\Delta Z_{(x,y)}$ é calculado pela Equação 6.5.

$$\begin{aligned} d &= \sqrt{W^2_{VOL} + H^2_{VOL} + D^2_{VOL}} \\ \Delta Z_{(x,y)} &= \frac{1}{d} \cdot \left(\frac{d}{2} - B_{(x,y)}.Z_T \right) \end{aligned} \quad (6.5)$$

Outras opções de iluminação são utilizar apenas $\Delta Z_{(x,y)}$ (*Depth Shading*) ou apenas $cos\,\theta_{(x,y)}$ (simplificação extrema do modelo de Phong, considerando apenas reflexão difusa com $K_d = 1$) como multiplicador de Y. É útil também, especialmente ao inspecionar

cortes planares com todos os objetos opacos, não aplicar iluminação alguma. Tanto neste caso como no depth shading, o estágio de estimativa de normais não precisa ser executado.

A vista resultante pode ter dimensão muito pequena para apresentação direta ao usuário (a dimensão de uma fatia de ressonância varia de 128×128 a 512×512, enquanto que a dimensão das telas gráficas da maioria das estações de trabalho variam entre 800×600 e 1600×1200). A aplicação de uma transformada de escala no estágio de projeção exigiria um tamanho de máscara variável para o estágio de splatting. A solução adotada é compor a vista 2D com uma escala 2D com parâmetros inteiros; Para aplicar um fator de ampliação de $2\times$, por exemplo, cada pixel projetado é pintado como um retângulo de 2×2 pixels na imagem da vista.

6.7 Exemplos

A Figura 6.7 mostra renderizações do volume não segmentado (a tonalização usa apenas os valores de intensidade do volume, pois ainda não há rótulos), com uma caixa de corte (a), exemplos de renderização de um volume segmentado (b–c), renderizações com diferentes modelos de iluminação (d), e (e) ilustra os efeitos da variação dos parâmetros da Equação 6.4.

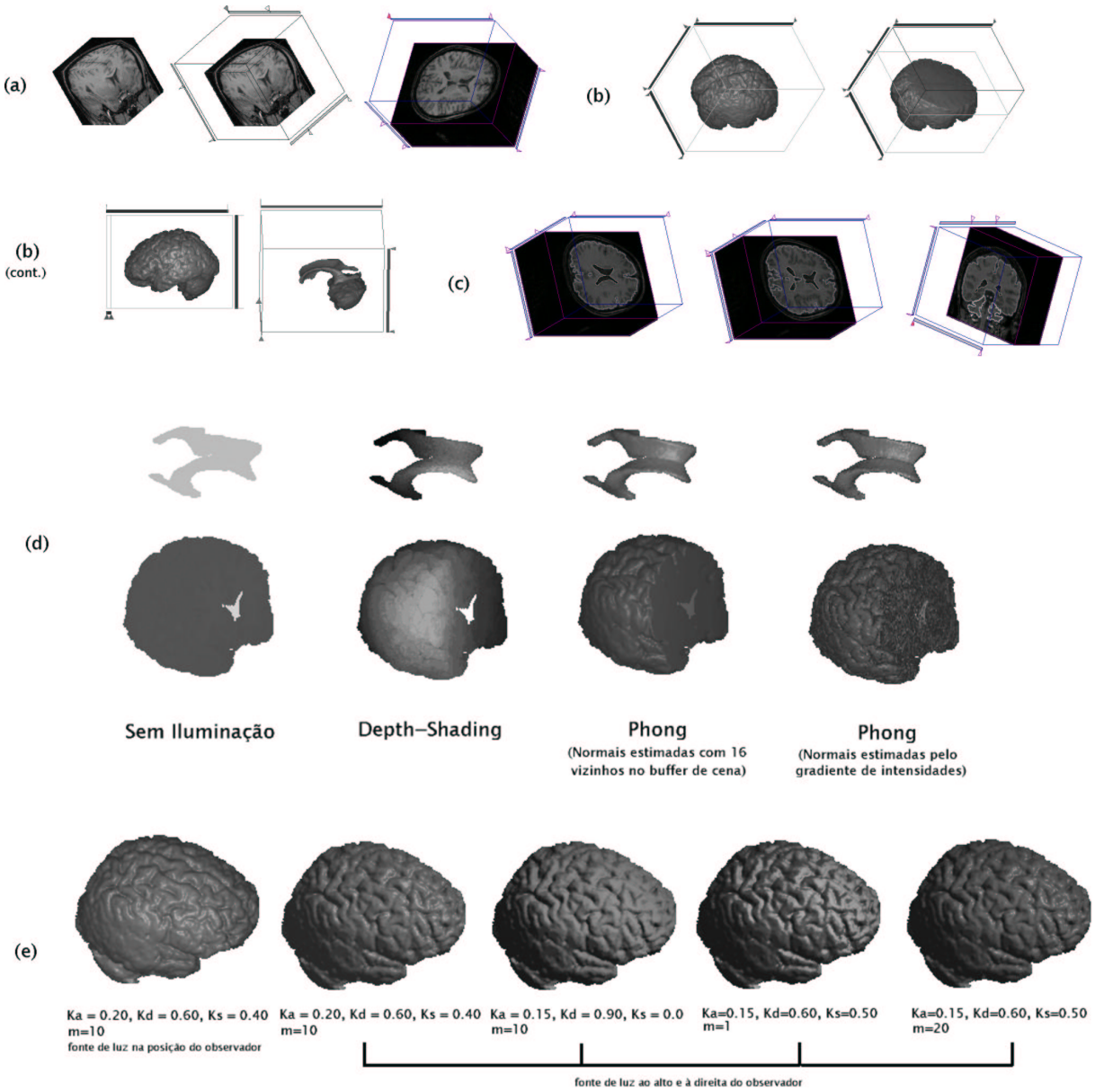


Figura 6.7: (a) Renderização opaca (sem segmentação); (b) Renderização de objetos; (c) Renderização de bordas; (d) Exemplos dos diversos métodos de iluminação. A estimativa de normal por gradientes mostra a falha do método em planos de corte; (e) Exemplos de renderizações com variações nos parâmetros de iluminação. $\alpha = 0.25$ e $\beta = 1.25$ em todas as renderizações.

6.8 Performance

Na prática, o estágio de projeção requer tempo proporcional ao número de voxels que satisfazem a condição de visibilidade (linha 4 do Algoritmo 6-1). Embora o algoritmo precise varrer todos os voxels do volume, o cálculo da transformada 3D (linha 5) e a comparação das profundidades (linha 6) tornam o processamento dos voxels "projetáveis" muito mais custoso que a simples varredura do laço da linha 3. O gráfico na Figura 6.8 mostra os tempos do estágio de projeção para 3 cenas de tamanhos diferentes, com a variação da condição de visibilidade (marcando objetos como visíveis ou não, alterando assim o número de voxels "projetáveis"), obtidas em um computador com processador Athlon de 1100 MHz.

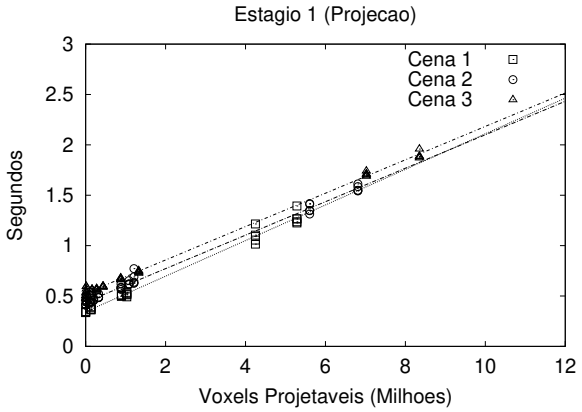


Figura 6.8: Tempo de execução do estágio de projeção para 3 cenas. Tamanhos: Cena 1: 5 285 952 voxels; Cena 2: 6 811 875 voxels; Cena 3: 8 350 290 voxels. As linhas indicam a regressão linear para cada cena.

Os estágios seguintes dependem apenas do tamanho do buffer de cena (2D) e do número de pixels onde há voxels projetados. A Tabela 6.1 mostra os tempos requeridos em cada estágio para vistas típicas de cenas de segmentação do cérebro (em um PC Athlon 1100 MHz). A Figura 6.9 mostra as 3 vistas testadas, para a cena 2. Para obter o corte opaco usamos uma variação do algoritmo de projeção em que apenas as faces da caixa de corte são varridas, já que o interior não será visível.

Nas modalidades não-opacas, onde é necessário varrer todo o volume, o principal componente do tempo total total de renderização é o estágio de projeção. Com o método de renderização apresentado, obtemos tempos médios em torno de 1 segundo ao longo de uma sessão de segmentação em um PC de 1100 MHz.

Realizamos, ao longo deste trabalho, diversas séries de segmentações para testar os

Descrição	Voxels Projetáveis	Proj. (segs.)	Splat. (segs.)	Est.N. (segs.)	Tonal. (segs.)	Total (segs.)
Cena 1, Objetos	1 037 881	0,4825	0,0079	0,1104	0,0214	0,6222
Cena 2, Objetos	1 207 146	0,6109	0,0089	0,1273	0,0233	0,7704
Cena 3, Objetos	1 327 394	0,6953	0,0094	0,1267	0,0248	0,8562
Cena 1, Bordas	2 642 976	0,6312	0,0201	0,0004	0,1584	0,8101
Cena 2, Bordas	3 425 625	0,8144	0,0278	0,0005	0,2028	1,0455
Cena 3, Bordas	4 175 145	0,9704	0,0265	0,0006	0,2423	1,2398
Cena 1, Corte Opaco	127 610	0,0827	0,0202	0,0004	0,1448	0,2893
Cena 2, Corte Opaco	148 350	0,0999	0,0285	0,0004	0,1860	0,3148
Cena 3, Corte Opaco	167 376	0,0928	0,0318	0,0006	0,2254	0,3506

Tabela 6.1: Consumo de tempo em cada estágio de renderização

métodos apresentados nos capítulos anteriores. Selecionamos a série mais longa (50 sessões de segmentação, realizando 5 segmentações distintas em 10 cenas do cérebro) e apresentamos na Tabela 6.2 algumas medições das renderizações realizadas. Embora as 10 cenas variem de tamanho, entre 5 285 952 e 8 994 433 voxels, o tempo de visualização depende fortemente da segmentação da cena e dos parâmetros de renderização (que definem quais voxels são projetáveis). O gráfico na Figura 6.10 mostra as medidas de tempo em função do tamanho da cena. Embora a regressão linear seja válida, é visível que a variação do tempo para um mesmo tamanho de cena (espalhamento vertical dos pontos plotados) é mais relevante que a variação com o tamanho da cena.

Tarefa	Contagem	(%)	Média (seg.)	Mín. (seg.)	Máx. (seg.)
Renderizações Opacas	601	(8,9%)	0,3123	0,2255	0,6149
Renderizações de Objetos	4954	(73,4%)	0,9425	0,2049	2,3590
Renderizações de Bordas	1190	(17,6%)	1,0730	0,2772	1,8793
Todas as Renderizações	6745	(100,0%)	0,9094	0,2049	2,3590

Tabela 6.2: Medidas de performance em 50 sessões de segmentação.

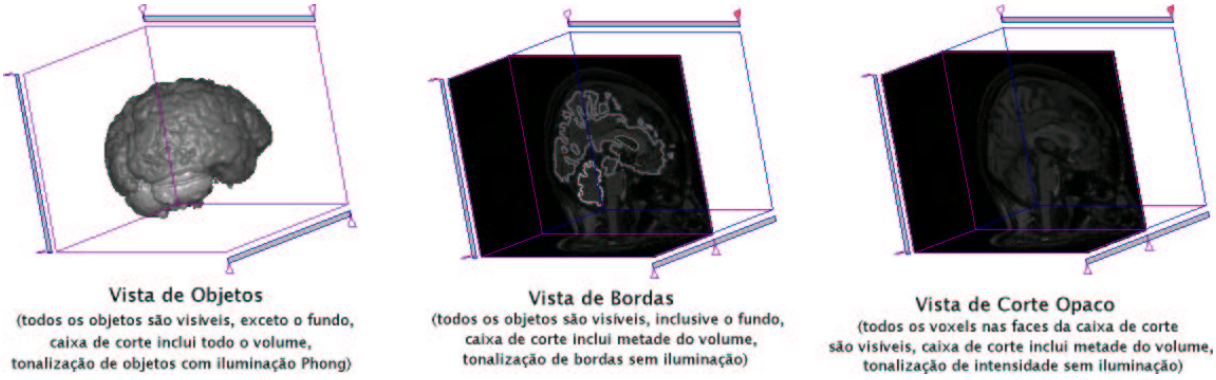


Figura 6.9: Parâmetros de vista usados nos testes da tabela 6.1 e as vistas obtidas para a cena 2.

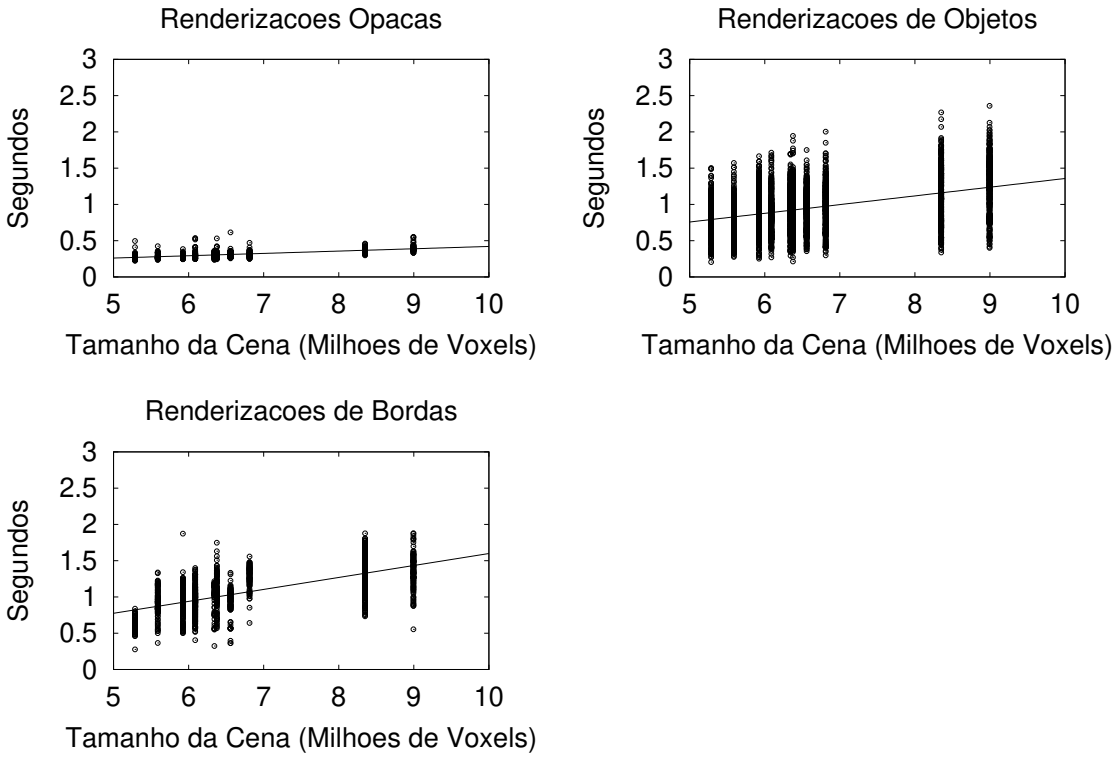


Figura 6.10: Tempo de renderização plotado em função do tamanho da cena, e sua regressão linear, para as três modalidades de renderização utilizadas.

6.9 Comentários

O método de estimativa de normal apresentado na seção 6.5 foi formulado independentemente durante este trabalho, mas a idéia de usar uma estrutura 2D, proveniente de um passo inicial de projeção, em geral um Z-buffer, é apresentada em [8], usando 4 vizinhos. O uso de um Z-buffer, entrentanto, discretiza as coordenadas X e Y no espaço transformado, causando artefatos na imagem renderizada.

Outros métodos encontrados na literatura para obter normais de superfícies são o uso do gradiente [34, 48] e a aproximação dos objetos segmentados por uma malha de polígonos. O uso do gradiente é muito dependente das características de intensidade da imagem, e pode não oferecer resultados adequados para exibir estruturas anatômicas sem constraste nas bordas (limites definidos por classificação geométrica). Além disso, o cálculo do gradiente precisa ser realizado no espaço tridimensional da cena, uma desvantagem em relação à estimativa sobre o espaço bidimensional da vista. O gradiente também não funciona em cortes por planos arbitrários, que precisam ser tratados como casos especiais dentro do processo de renderização. A redução a uma malha de polígonos [35, 28] pode ser lenta, e em geral inadequada para exibir imagens intermediárias durante a segmentação, uma vez que a identidade dos voxels no espaço da cena 3D é perdida, dificultando a avaliação da segmentação pelo usuário.

Em [43] os autores discutem qualitativamente diversas abordagens para a renderização de volumes, e [54] são discutidas abordagens para a estimativa de normais em volumes.

.

Capítulo 7

Conclusões

"Ah, há, há! Processo e resultado são coisas independentes. A realidade é complicada."
– Seta-san em *Love Hina*, de Ken Akamatsu.

7.1 Discussão

Imagens digitais tridimensionais, hoje usadas em larga escala na Medicina, são normalmente segmentadas fatia por fatia, devido ao alto custo computacional de aplicar algoritmos de segmentação e filtros sobre todo o volume. Com as ferramentas disponíveis hoje, leva-se de 10 a 20 minutos para segmentar um único objeto em um volume com 30 fatias.

Neste trabalho, atacamos o problema da segmentação interativa de volumes com um algoritmo que computa IFTs de forma diferencial, a DIFT. A IFT reduz problemas de processamento de imagens à computação de uma árvore de caminhos mínimos sobre um grafo, permitindo implementações eficientes de diversos operadores de processamento de imagens. A DIFT permite que árvores sejam adicionadas e removidas da floresta de forma diferencial, com tempo proporcional apenas ao tamanho da região afetada.

Instanciamos dois métodos eficazes de segmentação – Watershed e Conexidade Fuzzy Relativa – através da DIFT e mostramos que a DIFT viabiliza a segmentação interativa de volumes em PCs comuns, um requisito essencial para que o uso de aplicações de processamento de imagens médicas se popularize em ambientes clínicos.

A DIFT permitiu a implementação de um software aplicativo de segmentação em que o usuário espera tipicamente menos de 5 segundos para visualizar o resultado de suas

operações. A realização de tais segmentações com operadores semelhantes, baseados na IFT, levaria o usuário a esperar 20 segundos ou mais por cada resposta, tornando a segmentação de volumes impraticável. Os experimentos foram realizados com volumes obtidos em exames no Hospital das Clínicas da Unicamp, com as mesmas dimensões e características dos volumes de ressonância magnética que hospitais e clínicas usam no dia-a-dia (Todos volumes de alta resolução, com mais de 100 fatias, que em geral não são segmentados devido ao longo tempo que seria necessário com os métodos e ferramentas até então disponíveis).

Embora cada correção realizada com a DIFT possa, no pior caso, levar o mesmo longo tempo requerido pela primeira iteração, os experimentos realizados mostraram que o número de voxels afetados pelas correções é muito inferior ao número total de voxels do volume, levando a correções muito mais rápidas que a primeira iteração da DIFT.

Os experimentos realizados indicaram também que grande parte do tempo gasto para realizar uma segmentação interativa é gasta com a interação do usuário, que leva um tempo considerável para verificar a corretude da segmentação ao longo de todo o volume. Isto indica que o aperfeiçoamento da interface com o usuário pode contribuir consideravelmente para a redução do tempo exigido para completar a segmentação.

Conseguimos segmentar de 1 a 4 objetos simultaneamente em sessões de segmentação que variaram de 7 a 26 minutos, com tempos de resposta abaixo de 3 segundos na maioria da interações, processando volumes de forma realmente tridimensional (em vez processá-los fatia por fatia) em PCs comuns, sem qualquer hardware especializado. É um resultado gratificante, que demonstra o potencial da DIFT para a segmentação interativa de volumes. Podemos concluir que a DIFT viabiliza a segmentação de volumes grandes (> 100 fatias) em tempo menor ou igual àquele requerido para a realização da segmentação manual em volumes menores (que é a prática corrente).

7.2 Sugestões para Trabalhos Futuros

Pré-Processamento para Aplicações Específicas. Neste trabalho usamos técnicas simples de pré-processamento para os objetos de interesse, uma vez que o nosso foco estava na avaliação dos operadores baseados na DIFT. O estudo mais aprofundado das imagens e objetos a serem segmentados, em conjunto com o uso de algoritmos mais refinados para o pré-processamento, são um passo natural para a viabilizar a introdução de ferramentas baseadas na DIFT em ambientes clínicos.

Avaliações de corretude. Nos experimentos realizados, selecionamos objetos com bordas bem definidas, de forma que a corretude da segmentação pudesse ser verificada de

forma imediata pelo usuário. A ausência de bancos de dados públicos de segmentações em imagens volumétricas de alta resolução (cortes de 1mm ou menos) inviabilizou qualquer aferição objetiva das segmentações. O uso clínico da segmentação interativa de volumes depende de estudos que demonstrem a confiabilidade das segmentações obtidas.

Automatizar o que ainda é manual. Métodos de pré-processamento automático, escolha automática de sementes e de escolha automática dos parâmetros do DIFT-CFR podem ser de grande utilidade para reduzir o tempo requerido para completar a segmentação, além de reduzir a variabilidade de resultados entre diferentes usuários.

Desenvolver novos operadores de segmentação. Neste trabalho apresentamos dois operadores de segmentação baseados na função f_{max}, mas a IFT (e, por extensão, a DIFT) é uma ferramenta extraordinária para a experimentação com novos operadores.

Melhorar a interface com o usuário. Ao desenvolver a ferramenta IVS, não nos preocupamos muito com o projeto da interface com o usuário, e utilizamos apenas os mecanismos de interação presentes em qualquer PC (mouse e teclado). O projeto de uma interface consistente e eficiente é essencial para a popularização de softwares de processamento de imagens na comunidade médica.

.

Apêndice A

Estruturas de Dados

Neste apêndice apresentamosimplementações de duas estruturas de dados importantes para a IFT (Capítulo 2) e para a DIFT (Capítulo 3).

A.1 Implementação da Fila de Prioridades

Para ser utilizada pela IFT uma fila de prioridades deve prover pelo menos as seguintes 5 operações:

Insere(elemento, custo) – Insere um elemento na fila, com o custo dado.

Remove-Mínimo – Remove da fila e retorna um elemento de custo mínimo.

Altera-Prioridade(elemento, c_0, c_1) – Altera a prioridade de um elemento (que necessariamente está na fila, com custo c_0) para c_1.

Verifica-Presença(elemento) – Determina se um elemento está inserido na fila.

Fila-Vazia – Determina se a fila está vazia.

A melhor implementação de fila de prioridades para a IFT que conhecemos até o momento é uma variação da fila de *buckets* proposta por Dial [13].

A fila é composta por um *array* [10] circular de $\Delta + 1$ buckets, e cada bucket contém uma lista duplamente ligada que contém os elementos inseridos na fila com o custo associado àquele bucket. O valor de Δ é o maior incremento possível da função de custo f, e representa o maior incremento possível no custo de um caminho obtido pela concatenação de um vértice ao caminho. No caso de uma função de custo aditiva (como a do contra-exemplo da Figura 2.3), Δ seria o maior custo de aresta presente no grafo. Note que os buckets precisam suportar apenas o intervalo de custos dos vértices simultaneamente

presentes na fila (os spels com hachura intermediária da Figura 2.2, por exemplo). Como a diferença entre custos destes spels é um valor no intervalo $[0, \Delta]$, são necessários $\Delta + 1$ buckets para enumerar todas as possibilidades simultâneas.

A variação em relação à fila de Dial apresentada em [2] é a alocação de todos os $|\mathcal{I}|$ nós de lista ligada de antemão, aumentando drasticamente a performance por que remove os custos escondidos de alocação e desalocação dinâmica de nós (que requer a travessia de listas de blocos pelo algoritmo de alocação do sistema operacional).

As operações requeridas pela IFT são implementadas da seguinte forma:

Algoritmo A-1 – INSERE

ENTRADA: elemento p, custo c

1. *Calcule o bucket b associado ao custo c.*
2. *Ligue o nó associado a p ao fim a da lista ligada de b*
3. *Incremente a contagem de elementos na fila.*

Algoritmo A-2 – REMOVE-MÍNIMO

SAÍDA: um elemento de custo mínimo

1. *A partir do bucket associado ao menor custo, percorra o array circular de buckets até encontrar um bucket b não vazio.*
2. *Remova um elemento p do início da lista ligada de b.*
3. *Decremente a contagem de elementos na fila.*
4. *Retorne o elemento p.*

Algoritmo A-3 – ALTERA-PRIORIDADE

ENTRADA: elemento p, custos c_0 e c_1.

1. *Calcule o bucket b_0 associado ao custo c_0.*
2. *Remova o nó associado a p da lista ligada do bucket b_0.*
3. *Decremente a contagem de elementos na fila.*
4. *Chame* INSERE*(p, c_1).*

Algoritmo A-4 – VERIFICA-PRESENÇA

ENTRADA: elemento p.
SAÍDA: booleano, indicando presença do elemento na fila.

1. *Se o nó associado a p estiver em alguma lista (apontadores de elemento anterior e próximo elemento diferentes de nil), retorne verdadeiro, senão retorne falso.*

Algoritmo A-5 – FILA-VAZIA

SAÍDA: booleano, indicando se a fila está vazia.

1. *Se a contagem de elementos for* 0*, retorne verdadeiro; senão, retorne falso.*

O cálculo do bucket associado a um custo é imediato, realizado com uma soma e uma operação de resto de divisão, portanto $\Theta(1)$. Inserir ou remover nós de uma lista duplamente ligada também é $\Theta(1)$, exigindo apenas a atualização de 4 ponteiros (nó anterior e próximo nó do próprio elemento, e nó anterior do próximo e próximo nó do anterior). INSERE, portanto, leva tempo $\Theta(1)$. O passo 1 de REMOVE-MÍNIMO pode requerer até $\Delta + 1$ iterações, pois pode ter que percorrer todos os buckets em busca do próximo bucket não vazio. Os outros passos são $\Theta(1)$, portanto REMOVE-MÍNIMO leva tempo $O(\Delta)$. ALTERA-PRIORIDADE, VERIFICA-PRESENÇA e FILA-VAZIA levam tempo $\Theta(1)$ cada uma.

A desvantagem desta implementação é o uso de memória, pois durante toda a operação a estrutura mantém alocados $\Delta + 1$ buckets (que contêm apontadores de cabeça e cauda de suas listas ligadas) e $|\mathcal{I}|$ nós de lista ligada, cada um com dois apontadores (próximo nó e nó anterior). A estrutura gasta, portanto, $\Theta(\Delta + |\mathcal{I}|)$ espaço de armazenamento. Na prática, para imagens 2D o valor de $|\mathcal{I}|$ é pequeno (comparado à capacidade de memória dos computadores em uso); E para imagens 3D, embora a ocupação de memória da fila de prioridades seja considerável, também é $\Theta(|\mathcal{I}|)$ o espaço de armazenamento necessário para manter a cena anotada, e o espaço requerido pela cena anotada é algumas vezes maior que o requerido pela fila (pois diversos mapas, de tamanho $|\mathcal{I}|$, são mantidos), portanto, como regra geral, o sistema que tiver memória suficiente para manter a cena anotada em memória principal provavelmente terá memória para alocar a fila de prioridades enquanto a IFT estiver sendo executada.

A.2 Implementação dos Conjuntos

A DIFT depende de diversas operações com conjuntos (inicialização, percurso, verificação de pertinência, união, subtração). Nesta seção discutimos uma implementação eficiente desta estrutura de dados para a DIFT. Na DIFT, os conjuntos contêm spels e são sempre subconjuntos de $\mathcal{I}$. Podemos assumir que cada spel é um inteiro entre 0 e $|\mathcal{I}| - 1$.

Para representar um conjunto $S \subseteq \mathcal{I}$, mantemos duas estruturas auxiliares: um vetor de booleanos S_V e uma lista duplamente ligada de inteiros S_L. Cada posição de S_V armazena a pertinência de cada elemento de $\mathcal{I}$ em S. A lista S_L contém, em ordem arbitrária, sem repetições, todos os elementos de S. As operações sobre o conjunto S podem ser implementadas da seguinte forma:

Algoritmo A-6 – Cria-Conjunto

Entrada: Domínio $\mathcal{I}$.
Saída: Conjunto $S = \emptyset$.

1. *Aloque* $S_V[0 \ldots |\mathcal{I}| - 1]$ *e* S_L.
2. **Para Cada** $p \in [0, |\mathcal{I}|)$ **Faça** $S_V[p] \leftarrow 0$.
3. **Retorne** $S = \{S_V, S_L\}$.

A linha 2 pode ser implementada com operações de preenchimento de blocos com custo desprezível, portanto podemos assumir que um conjunto pode ser alocado em tempo $\Theta(1)$.

A pertinência de um elemento p a um conjunto $S = \{S_V, S_L\}$ pode ser determinada em tempo $\Theta(1)$ verificando $S_V[p]$. Um conjunto representado desta forma pode ser percorrido em tempo $|S|$ percorrendo a lista ligada S_L.

Algoritmo A-7 – Uniao

Entrada: Conjuntos $S = \{S_V, S_L\}$ e $T = \{T_V, T_L\}$.
Saída: Conjunto $S = S \cup T$.

1. **Para Cada** $p \in T$ **Faça**
2. ⌞ **Se** $p \notin S$ **Então** $S_V[p] \leftarrow 1$, *Insira* p *em* S_L.
3. **Retorne** $S = \{S_V, S_L\}$.

O procedimento Uniao percorre T colocando seus elementos em S (se ainda não presentes). O algoritmo leva tempo $\Theta(|T|)$. A união de um conjunto com um único elemento leva, portanto, $\Theta(1)$.

Algoritmo A-8 – SUBTRAI-CONJUNTO

ENTRADA: Conjuntos $S = \{S_V, S_L\}$ e $T = \{T_V, T_L\}$.
SAÍDA: Conjunto $S = S \setminus T$.

1. **Para Cada** $p \in T$ **Faça**
2. $\quad$ **Se** $p \in S$ **Então** $S_V[p] \leftarrow 0$.
3. **Para Cada** $p \in S_L$ **Faça**
4. $\quad$ **Se** $S_V[p] = 0$ **Então** *remova* p *de* S_L.
5. **Retorne** $S = \{S_V, S_L\}$.

O primeiro laço percorre T e zera em S_V os elementos pertencentes a $T \cap S$. O segundo laço percorre S_L removendo desta lista os elementos pertencentes a $T \cap S$. O algoritmo leva tempo $\Theta(|S| + |T|)$ para computar $S \setminus T$.

Note que remover um determinado elemento p de um conjunto S é um caso particular de SUBTRAI-CONJUNTO, em que $T = \{p\}$, e requer tempo $\Theta(|S|)$.

A remoção de um elemento qualquer de um conjunto S pode ser implementada em tempo $\Theta(1)$: obtém-se o primeiro elemento da lista S_L, zera-se a posição correspondente em S_V e remove-se o primeiro elemento de S_L.

A interseção entre dois conjuntos pode ser determinada pelo algoritmo abaixo:

Algoritmo A-9 – INTERSECAO

ENTRADA: Conjuntos $S = \{S_V, S_L\}$ e $T = \{T_V, T_L\}$.
SAÍDA: Conjunto $S = S \cap T$.

1. **Para Cada** $p \in T$ **Faça**
2. $\quad$ **Se** $p \in S$ **Então** $S_V[p] \leftarrow 0$.
3. **Para Cada** $p \in S_L$ **Faça**
4. $\quad$ **Se** $S_V[p] = 1$ **Então** *remova* p *de* S_L.
5. $\quad$ $S_V[p] \leftarrow \neg S_V[p]$.
6. **Retorne** $S = \{S_V, S_L\}$.

O primeiro laço zera a pertinência S_V dos elementos em $S \cap T$. O segundo laço remove da lista S_L os elementos com $S_V = 1$, ou seja, $S \setminus T$, e inverte as pertinências S_V dos elementos em S (elementos em $S \setminus T$ vão de 1 para 0, e elementos em $S \cap T$ vão de 0 para 1), corrigindo-as. Este algoritmo calcula $S \cap T$ em tempo $\Theta(|S| + |T|)$.

.

Apêndice B

Estruturas Anatômicas

Neste Apêndice apresentamos a nomenclatura e localização das estruturas anatômicas (ver Figuras B.1, B.2 e B.3) e tecidos citados no texto.

CSF Sigla comum na literatura em inglês, de *Cerebrospinal Fluid*, é o fluido cérebro-espinhal, que envolve o cérebro e a medula espinhal.

WM Sigla para *White Matter*, a Substância Branca do cérebro.

GM Sigla para *Gray Matter*, a Substância Cinzenta do cérebro, e que forma o córtex.

LV Ventrículos Laterais (em inglês, *Lateral Ventricles*).

CN Núcleos Caudados (2), estruturas adjacente aos ventrículos laterais.

Putamen Estruturas (2) próximas aos Núcleos Caudados, muitas vezes conexas a eles e de intensidade de voxel semelhante (por serem formados de GM, como os núcleos caudados).

Cerebelo Em inglês, *cerebellum*.

Pons Bulbo que, em imagens de ressonância magnética em modalidade T1, aparece indiferenciável (exceto pelo contorno característico) da medula.

Medula Em inglês, *medulla*.

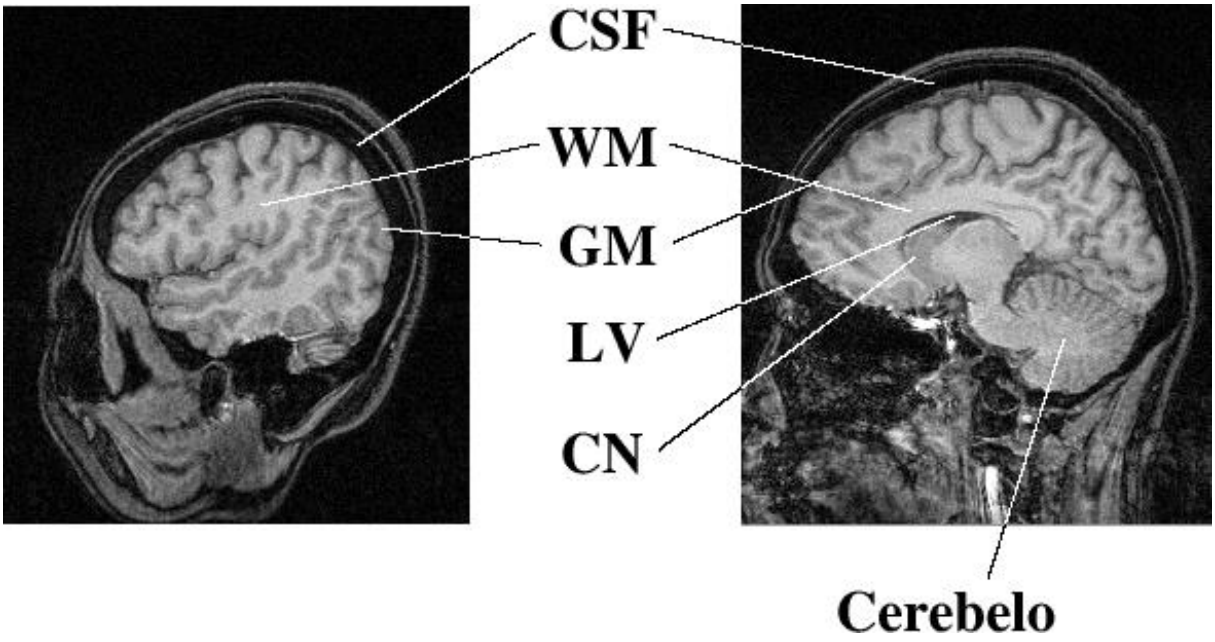


Figura B.1: Estruturas anatômicas em cortes sagitais.

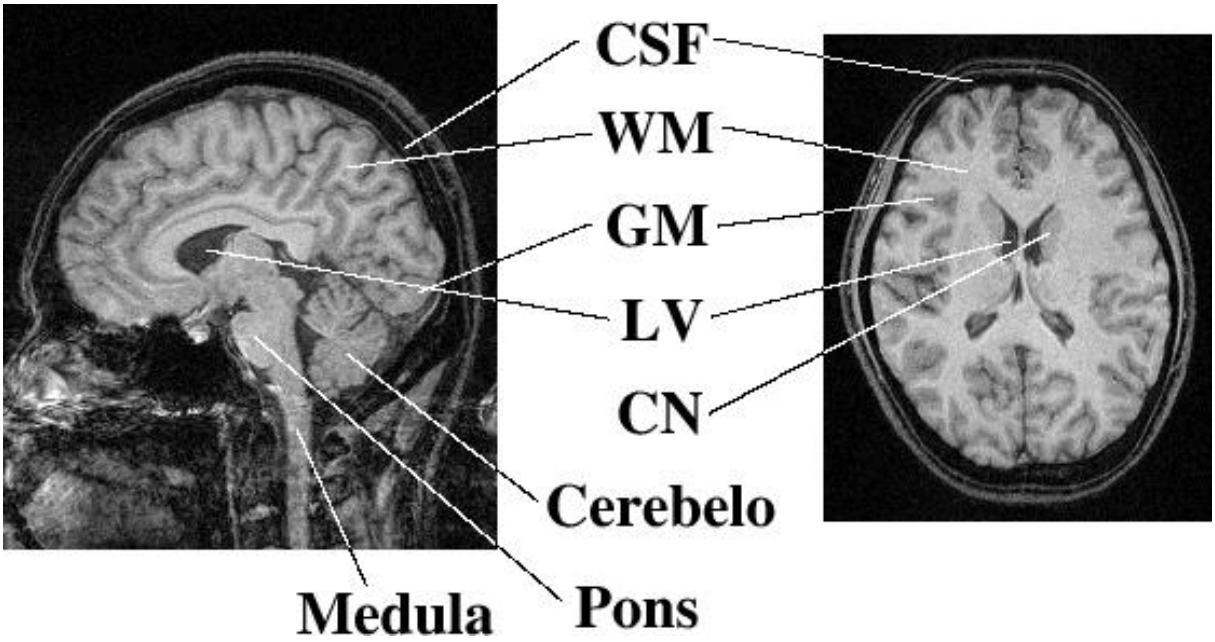


Figura B.2: Estruturas anatômicas em cortes sagital (esq.) e transversal (dir.).

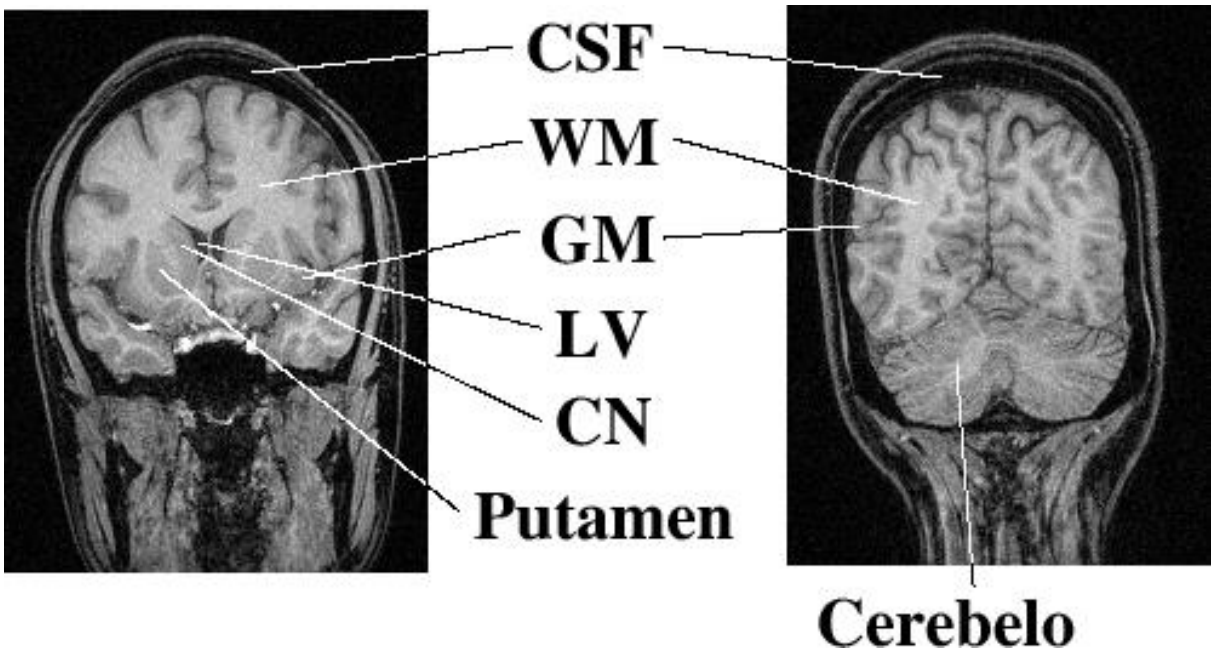


Figura B.3: Estruturas anatômicas em cortes coronais.

.

Bibliografia

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

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

[3] S. Ando. Consistent gradient operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):252–265, 2000.

[4] F.P.G. Bergo and A.X. Falcão. Interactive 3D segmentation of brain MRI with differential watersheds. In *III Simpósio Catarinense de Processamento Digital de Imagens*, pages 89–96, Florianópolis, Brasil, October 2003.

[5] F.P.G. Bergo and A.X. Falcão. *IVS – Interactive Volume Segmentation – Quick Guide*. November 2003. Available from http://www.ic.unicamp.br/%7Eafalcao/ivs.

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

[7] G. Bueno, O. Musse, F. Heitz, and J.P. Armspach. Three-dimensional segmentation of anatomical structures in MR images on large data bases. *Magnetic Resonance Imaging*, 19:73–88, 2001.

[8] L. Chen, G.T. Herman, R.A. Raynolds, and J.K. Udupa. Surface shading in the cuberille environment. *IEEE Computer Graphics and Applications*, 5(12):33–43, December 1985.

[9] C.A. Cocosco, A.P. Zijdenbos, and A.C. Evans. Automatic generation of training data for brain tissue classification from MRI. In *5th MICCAI, Lecture Notes in Computer Science 2488*, pages 516–523, Tokyo, 2002. Springer-Verlag.

[10] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, New York, NY, 1991.

[11] B.S. da Cunha. Projeto de operadores de processamento e análise de imagens usando a transformada imagem-floresta. Master's thesis, Instituto de Computação - UNICAMP, June 2001.

[12] M. Das and J. Anand. Robust edge detection in noisy images using an adaptive stochastic gradient technique. In *International Conference on Image Processing (ICIP)*, pages 2149–2152, Washington, D.C., 1995.

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

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

[15] J.S. Duncan and N. Ayache. Medical image analysis: Progress over two decades and the challenges ahead. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):85–105, 2000.

[16] A.X. Falcão. *Paradigmas de Segmentação de Imagens Guiada Pelo Usuário: Livewire, Live-Lane e 3D-Live-Wire*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, SP, 1996.

[17] A.X. Falcão and F.P.G. Bergo. The iterative image foresting transform and its application to use r-steered 3D segmentation. In *Proc. of SPIE on Medical Imaging*, volume 5032, pages 1464–1475, Feb 2003.

[18] A.X. Falcão, L.F. Costa, and B.S. da Cunha. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, 2002.

[19] A.X. Falcão and B. S. da Cunha. Multiscale shape representation by the image foresting transform. In *Proceedings of SPIE on Medical Imaging*, volume 4322, pages 1091–1100, San Diego, CA, February 2001.

[20] A.X. Falcão, B. S. da Cunha, and R. A. Lotufo. Design of connected operators using the image foresting transform. In *Proceedings of SPIE on Medical Imaging*, volume 4322, pages 468–479, San Diego, CA, February 2001.

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

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

[23] P. Felkel, M. Bruckschwaiger, and R. Wegenkittl. Implementation and complexity of the watershed-from-markers algorithm computed as a minimal cost forest. *Computer Graphics Forum (Eurographics)*, 20(3):C26–C35, 2001.

[24] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practice.* Addison-Wesley, New York, second ed. edition, 1990.

[25] R.C. Gonzales and R.E. Woods. *Digital Image Processing.* Addison-Wesley, 1992.

[26] W.E. Higgins and E.J. Ojard. Interactive morphological watershed analysis for 3D medical images. *Computerized Medical Imaging and Graphics*, 17(4/5):387–395, 1993.

[27] K.H. Höhne and W.A. Hanson. Interactive 3D-segmentation of MRI and CT volumes using morphological operations. *Journal of Computer Assisted Tomography*, 16(2):285–294, 1992.

[28] M. Jackowski, M. Satter, and A. Goshtasby. Approximating digital 3D shapes by rational Gaussian surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):56–69, 2003.

[29] M. Kamber, R. Shingal, D. Collins, G. Francis, and A. Evans. Model-based 3-D segmentation of multiple sclerosis lesions in magnetic resonance brain images. *IEEE Transactions on Medical Imaging*, 14(3):442–453, March 1995.

[30] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[31] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of viewing transformation. *Computer Graphics*, pages 451–458, 1994.

[32] R.J. Lapeer, A.C. Tan, and R. Aldridge. Active watersheds: Combining 3D watershed segmentation and active contours to extract abdominal organs from MR images. In *5th MICCAI, Lecture Notes in Computer Science 2488*, pages 596–603, Tokyo, 2002. Springer-Verlag.

[33] T. Lei, J.K. Udupa, P.K. Saha, and D. Odhner. Artery-vein separation via MRA - An image processing approach. *IEEE Transactions on Medical Imaging*, 20(8), 2001.

[34] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, pages 29–37, May 1988.

[35] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer & Graphics*, pages 163–169, July 1987.

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

[37] R.A. Lotufo, A.X. Falcão, and F.A. Zampirolli. IFT-Watershed from gray scale marker. In *Proceedings of XV Brazilian Symposium on Computer Graphics and Image Processing*, Fortaleza, CE, Oct 7–10 2002. in press.

[38] U. Manber. *Introduction to algorithms: A creative approach*. Addison Wesley, New York, 1989.

[39] T. McInerney and D. Terzopoulos. T-snakes: Topology adaptive snakes. *Medical Image Analysis*, 4:73–91, 2000.

[40] N. Moon, E. Bullitt, K. van Leemput, and G. Gerig. Automatic brain and tumor segmentation. In *5th MICCAI, Lecture Notes in Computer Science 2488*, pages 372–379, Tokyo, 2002. Springer-Verlag.

[41] L.G. Nyúl, A.X. Falcão, and J.K. Udupa. Fuzzy-connected 3D image segmentation at interactive speeds. In *Proceedings of SPIE on Medical Imaging*, volume 3979, pages 212–223, San Diego, CA, February 2001.

[42] R. Pohle, T. Behlau, and K.D. Toennies. Segmentation of 3-D medical image data sets with a combination of region based initial segmentation and active surfaces. In *Proc. of SPIE on Medical Imaging*, volume 5032, pages 1225–1232, Feb 2003.

[43] Jonathan C. Roberts. An overview of rendering from volume data — including surface and volume rendering. Technical Report 13-93*, University of Kent, Canterbury, UK, December 1993.

[44] L.M. Rocha. Shell rendering com fatoração shear-warp. Master's thesis, Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, 2002.

[45] P. Saha and J.K. Udupa. Scale-based fuzzy connected image segmentation: Theory, algorithms and validation. *Computer Vision and Image Understanding*, 77(2):145–174, 2000.

[46] P.K. Saha and J.K. Udupa. Relative fuzzy connectedness among multiple objects: theory, algorithms, and applications in image segmentation. *Computer Vision and Image Understanding*, 82:42–56, 2001.

[47] A. Schenk, G. Prause, and H.O. Peitgen. Efficient semiautomatic segmentation of 3D objects in medical images. *Lecture Notes in Computer Science*, 1935:186–195, 2000.

[48] U. Tiede, K.H. Hoehne, M. Bomans, A. Pommert, M. Riemer, and G. Wiebecke. Surface rendering. *IEEE Computer Graphics and Applications*, 10(2):41–53, 1990.

[49] R.S. Torres, A.X. Falcão, and L.F. Costa. Shape description by image foresting transform. In *14th International Conference on Digital Signal Processing*, pages 1089–1092, Santorini, Greece, July 2002.

[50] J.K. Udupa and G.T. Herman. *3D Imaging in Medicine*. CRC Press, Boca Raton, FL, 2000.

[51] J.K. Udupa and S. Samarasekera. Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. *Graphical Models and Image Processing*, 58:246–261, 1996.

[52] K. van Leemput et al. Automated model-based tissue classification of MR images of the brain. *IEEE Transactions on Medical Imaging*, 18(10):897–908, October 1999.

[53] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991.

[54] R. Yagel, D. Cohen, and A. Kaufman. Normal estimation in 3D discrete space. *The Visual Computer*, 8(5-6):278–291, 1992.