

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

**Paradigmas de Segmentação de Imagens Guiada
pelo Usuário: *Live-Wire*, *Live-Lane* e *3D-Live-Wire***

Autor: **Alexandre Xavier Falcão**

Orientador: **Prof. Dr. Roberto de Alencar Lotufo**

Co-orientador: **Prof. Dr. Jayaram K. Udupa**

Tese submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Doutor em Ciências.

11 de Maio de 1997

Resumo

Este trabalho trata o problema de segmentação de imagens em situações onde os métodos automáticos falham requerendo extensiva assistência do usuário no processo. Como solução são propostos três paradigmas de segmentação interativa, denominados *live-wire*, *live-lane* e *3D-live-wire*, que têm como metas: (i) prover ao usuário, **tão completo quanto for possível**, o controle sobre o processo de segmentação **enquanto** este estiver em andamento e (ii) minimizar o envolvimento do usuário e o tempo total do usuário requeridos para segmentação. Os métodos podem ser usados para extrair objetos 2D, 3D e 4D de dados de imagem multidimensionais (e.g. imagens tomográficas). A estratégia usada pelos métodos é a de explorar a superior habilidade do operador humano (comparada com a dos algoritmos de computador) em reconhecer o objeto desejado na imagem e a superior habilidade dos algoritmos de computador (comparada com a do operador humano) em delinear este objeto. Os métodos oferecem três mecanismos de interação com o usuário, a segmentação é feita delineando cada borda 2D de um dado objeto de interesse fatia por fatia e a escolha entre os métodos depende da necessidade de envolvimento do usuário com o processo. Em *live-wire* e *live-lane*, o usuário interage ativamente na seleção de segmentos ótimos de borda em cada fatia. O método *live-lane* oferece ao usuário maior controle sobre o processo do que o *live-wire*, mas requer maior envolvimento. O método *3D-live-wire* requer o mínimo de envolvimento comparado com os outros métodos. Em *3D-live-wire*, o usuário seleciona inicialmente poucos cortes ortogonais ao plano das fatias e usa o método *live-wire* para traçar em cada corte uma borda 2D de um dado objeto 3D de interesse. Em seguida, pontos são gerados sobre a borda 3D do objeto que é delineada automaticamente fatia por fatia. Os métodos são avaliados em função da velocidade e repetibilidade do processo e tendo como base de comparação o traçado manual. Os métodos *live-wire* e *live-lane* são mais repetitivos e 1,5-2,5 vezes mais rápidos do que o traçado manual. O método *3D-live-wire* é o mais repetitivo de todos, 2-6 vezes mais rápido do que *live-lane* e 3-15 vezes mais rápido do que o traçado manual.

Abstract

In multidimensional imaging, there are, and will continue to be, situations wherein automatic image segmentation methods fail and extensive user assistance in the process is needed. For such situations, we propose a novel *user-steered* image boundary segmentation paradigm under three new methods, *live-wire*, *live-lane* and *3D-live-wire*, where the main goals are: (i) to provide as *complete a control* as possible to the user on the segmentation process *while* it is being executed, and (ii) to minimize the user involvement and the total user's time required for segmentation. The strategy to reach these goals is to exploit the synergy between the superior abilities of human observers (compared to computer algorithms) in object recognition and of computer algorithms (compared to human observers) in object delineation. The methods are designed to delineate 2D boundaries of 2D, 3D and 4D objects in a slice-by-slice fashion. The methods provide three mechanisms of user interaction and the best choice among them depends on the need of user involvement. In *live-wire* and *live-lane*, the user actively interacts in the selection process of optimum boundary segments in each slice. The method *live-lane* allows more user control than *live-wire* although it requires more user involvement. The method *3D-live-wire* requires a minimum of user involvement compared to the others. In *3D-live-wire*, the user initially selects a few orthogonal slices and traces a 2D boundary of a given 3D object in each orthogonal slice. Afterwards, there is a sufficient number of points on the 3D boundary of the object to do *live-wiring* automatically on all axial slices. We describe evaluation studies to compare the utility of the new methods with that of manual tracing based on speed and repeatability of tracing and on data taken from a large on-going application. We conclude that *live-wire* and *live-lane* are more repeatable and 1.5-2.5 times faster than manual tracing, and *3D-live-wire* is more repeatable than the others, 2-6 times faster than *live-lane* and 3-15 times faster than manual tracing.

Agradecimentos

Eu gostaria de agradecer aos professores Dr. Roberto de A. Lotufo e Dr. Jayaram K. Udupa pelo apoio técnico, motivação e orientação durante o desenvolvimento deste trabalho; ao colega Supun Samarasekera pelo apoio técnico; aos amigos Ernesto, Margarida, Gonzaga, Jussara, Miguel, Junior, Cida, Miki, Mara, Lougang e Cyro pelo companheirismo, motivação e ajuda nas horas difíceis; e aos meus familiares, principalmente minha mãe, avó, irmã, cunhado e sogros pela motivação e confiança depositadas. Um agradecimento especial à minha esposa Paula pela motivação, paciência, amizade, apoio e amor incondicional dedicados em todas as situações.

...à minha esposa Paula Regina Kuser Falcão.

Sumário

RESUMO	i
ABSTRACT	ii
AGRADECIMENTOS	iii
LISTA DE FIGURAS	iii
LISTA DE TABELAS	viii
1 Introdução	1
1.1 Motivação	2
1.2 Conceitos Básicos	4
1.3 Métodos Propostos	7
1.3.1 <i>Live-Wire, Live-Lane e 3D-Live-Wire</i>	8
1.3.2 Metodologia de Avaliação	10
1.4 Comparação com Outros Métodos	11
1.5 Estrutura do Trabalho	13
2 Classificação e Conectividade	15
2.1 Classificação	17
2.1.1 Seleção de Características de Imagem	18
2.1.2 Geração de Custos	20
2.1.3 Treinamento	22

2.1.4	Comentários	25
2.2	Conectividade	28
2.2.1	Algoritmo LW_1	31
2.2.2	Algoritmo LW_2	35
2.2.3	Comentários	39
2.3	Conclusão	40
3	<i>Live-Wire e Live-Lane</i>	44
3.1	<i>Live-Wire</i>	46
3.1.1	Exemplos e Propriedades	47
3.1.2	Outras Funções de Custo	51
3.1.3	Valor de Limiar	53
3.1.4	Região Anular	54
3.2	<i>Live-Lane</i>	56
3.2.1	Faixa de Largura Fixa	57
3.2.2	Faixa de Largura Variável	58
3.3	Avaliação	59
3.4	Conclusão	68
4	<i>3D-Live-Wire</i>	69
4.1	Metodologia	70
4.1.1	Seleção de Cortes	72
4.1.2	Ordenação de Vértices	76
4.1.3	Detecção de Borda	78
4.2	Avaliação	78
4.3	Conclusão	84
5	Conclusão e Discussão	86
	Referências Bibliográficas	90

Lista de Figuras

1.1	Imagem da fatia de um joelho obtida por CT. A figura indica o patela (osso da rótula do joelho) como exemplo de objeto 2D.	4
1.2	Processo de aquisição de imagens tomográficas resultando em uma cena 3D. .	5
1.3	As figuras indicam o patela como exemplo de objeto 3D. (a) Cena 3D formada pelo empilhamento das fatias de um joelho. (b) Visualização do interior da cena 3D através de um corte ortogonal ao plano das fatias.	6
1.4	Borda de um objeto em uma cena 2D definida como um contorno fechado, conectado e orientado formado por arestas de pixel.	7
1.5	Espaço de Segmentação.	14
2.1	A figura mostra a reconstrução 3D dos ossos do joelho após segmentação. (a) Segmentação usando apenas classificação por limiar. Insuficiente para separar o patela dos outros ossos. (b) Segmentação usando classificação por limiar seguida de conectividade. Apenas o patela é extraído da cena 3D.	16
2.2	Um elemento da borda (bel) é uma aresta de pixel orientada. Os quatro possíveis tipos de bels em uma cena 2D são mostrados. Para qualquer orientação da borda, um bel $b = (p, q)$ é sempre definido tal que p está à esquerda de b e q está à direita. Para uma borda orientada no sentido anti-horário, o pixel p está no interior do objeto e o pixel q no exterior.	17
2.3	Vizinhança de b usada para definir f_3, f_4, f_5 , e f_6	19
2.4	A figura ilustra o processo de treinamento da borda de uma secção transversal da traquéia. (a) Uma fatia transversal da traquéia de um paciente obtida por CT mostrando a região de interesse para treinamento. (b) Segmento da borda de interesse desenhado pelo usuário. (c) Imagem do custo conjunto gerada após a classificação usando as características f_1, f_2 e f_4	26

- 2.5 A figura mostra como o custo conjunto incorpora a informação sobre a orientação da borda. A orientação é assumida no sentido anti-horário e o custo conjunto é gerado usando as características f_{12}, f_{13} e f_{14} sensíveis à orientação da borda. (a) Fatia transversal da junta do pulso de um paciente obtida por MRI. (b) Imagem do custo conjunto gerada usando apenas os valores de custo de bels com orientação oeste e sul. Note que custos mais baixos são atribuídos às partes superior e lateral esquerda da borda. (c) Imagem do custo conjunto gerada usando apenas os valores de custo de bels com orientação leste e norte. Custos mais baixos são atribuídos às partes inferior e lateral direita da borda. 27
- 2.6 (a) Um grafo $G(V, E)$ definido para a cena \mathbf{C} após classificação. Um caminho $P = (e_1, e_2, \dots, e_5)$ em G entre os vértices v_0 e v_5 . (b) Cada vértice $v \in V$ define um conjunto $A(v) = u_1, u_2, u_3, u_4$ de no máximo 4 vértices vizinhos 4-adjacentes. 29
- 2.7 A figura ilustra o algoritmo LW_1 . (a) Um grafo de entrada representando uma cena \mathbf{C} após classificação, onde valores de custo conjunto de $[0,10]$ são associados aos bels de \mathbf{C} e o vértice inicial é representado pelo círculo escuro. As figuras de (b) a (f) ilustram respectivamente a situação do grafo de saída antes da primeira iteração, após a primeira iteração, após a segunda iteração, após a sétima iteração e após dezoito iterações. Nestas figuras, o próximo vértice a sair da fila é sempre representado pelo círculo parcialmente escuro. . 34
- 2.8 A figura ilustra o algoritmo LW_2 . (a) Um grafo de entrada representando uma cena \mathbf{C} após classificação, onde valores de custo conjunto de $[0,10]$ são associados aos bels de \mathbf{C} e o vértice inicial é representado pelo círculo escuro. As figuras de (b) a (f) ilustram respectivamente a situação do grafo de saída antes da primeira iteração, após a primeira iteração, após a segunda iteração, após a sétima iteração e após quinze iterações. Nestas figuras, o próximo vértice a sair da fila é sempre representado pelo círculo parcialmente escuro. . 38
- 2.9 (a) Uma fatia do joelho obtida por CT. (b) Uma imagem mostrando o custo conjunto de todos os bels da cena (a), usando a característica $f_6, c_2, \mu_2 = h_2 =$ (valor máximo de f_6), $\sigma_2 = (\text{valor máximo de } f_6)/10$ e $l_2 = 0$. (c) Caminho ótimo sendo atraído pelas bordas de alto contraste. (d) Treinamento para as bordas de baixo contraste. (e) Função de custo conjunto após treinamento. (f) Caminho ótimo sendo atraído pelas bordas de baixo contraste. (g) Caminho ótimo sendo repellido pelas bordas de alto contraste. 42
- 3.1 A figura mostra uma fatia do pé de um paciente obtida por MRI, onde o objeto em evidência é o osso tálus. O caminho ótimo falha quando tenta encontrar a borda através de um único segmento de custo mínimo calculado usando v_s e v_e como os vértices de um bel inicial b_0 . O critério de optimalidade leva o algoritmo a encontrar pequenas voltas na vizinhança de b_0 45

- 3.2 A figura ilustra o processo de segmentação usando *live-wire*. 46
- 3.3 A figura ilustra a segmentação via *live-wire* de uma fatia transversal da traquéia obtida por CT. (a) O primeiro segmento $P_1 = \langle v_0, v_1 \rangle$ de *live-wire*. (b) O segundo segmento $P_2 = \langle v_1, v_0 \rangle$ de *live-wire*. (c) O segmento $\langle v_s, v_e \rangle$ ilustra a tendência do *live-wire* optar por caminhos mais curtos mostrando que P_1 é o segmento mais longo possível que descreve a borda desejada partindo de v_0 48
- 3.4 (a) Fatia da junta do pulso obtida por MRI. O segmento de *live-wire* mostra-se imune ao tipo de ruído apresentado na borda em segmentação. (b) Fatia de um veia do pulso obtida por MRI. O segmento de *live-wire* mostra-se imune ao buraco apresentado na borda da veia. 49
- 3.5 A figura ilustra a importância das características sensíveis à orientação nos métodos *live*. (a) Uma fatia do pé de um paciente obtida por MRI, onde objeto desejado é o osso tálus. (b) Um segmento de *live-wire* é obtido incluindo apenas características independentes da orientação da borda. O segmento é atraído por partes espúrias de contraste similar na imagem. (c) A orientação da borda é assumida no sentido anti-horário. Um segmento de *live-wire* entre os mesmos dois vértices de (b) é obtido usando características sensíveis à orientação. A figura mostra uma região de interesse onde o segmento de contraste similar não atrai o *live-wire* visto que sua orientação não é favorável. (d) Um segmento de *live-wire* partindo do mesmo vértice inicial de (c), mas com orientação no sentido contrário, ilustra a sensibilidade do método à orientação previamente adotada para a borda. 50
- 3.6 Fatia da junta do pulso obtida por MRI. A figura ilustra a utilização de outras funções de custo para o caminho ótimo. (a) Segmento de *live-wire* utilizando como função de custo a soma do custo $c(b)$ de cada bel b deste segmento. (b) Segmento de *live-wire* partindo do mesmo vértice inicial v_s de (a) usando como função de custo a soma do custo $c(b)/d(b)$ de cada bel b , onde $d(b)$ é a distância Euclideana entre o vértice final de b e v_s . Note que o segmento em (b) é bem mais longo que em (a). 52
- 3.7 Fatia do pé obtida por MRI. A figura ilustra o efeito da função de custo $c(b)/d(b)$ na borda do osso tálus. (a) Segmento de *live-wire* desejado e obtido com a função de custo $c(b)$. (b) Segmento de *live-wire* obtido com a função de custo $c(b)/d(b)$. Note que o segmento em (b) é mais longo, mas adere a outras bordas da cena. 52

3.8	Fatia da junta do pulso obtida por MRI. A figura ilustra o efeito da utilização de um valor de limiar T nos algoritmos de conectividade. (a) Primeiro segmento de <i>live-wire</i> . (b) Segundo segmento de <i>live-wire</i> . Os segmentos ficam mais curtos e neste caso são necessários 11 segmentos para completar a borda. Em contra-partida, o tempo de processamento para cada vértice reduz 25 vezes.	54
3.9	Região anular de largura W da fatia anterior projetada na fatia atual. O contorno na fatia atual está dentro da região anular.	55
3.10	(a) Dois contornos da fatia atual não podem ser extraídos usando a região anular resultante de um único contorno da fatia anterior. (b) Duas regiões anulares resultantes de dois contornos na fatia anterior são projetadas na fatia atual. O contorno da fatia atual está totalmente inserido na região anular.	55
3.11	A figura ilustra o processo de segmentação usando <i>live-lane</i>	56
3.12	Uma fatia do pé de um paciente obtida por MRI mostrando as bordas dos ossos tálus e calcâneo usadas na avaliação dos métodos <i>live</i>	61
4.1	A figura mostra um objeto 3D, o qual é representado por duas estruturas (dois contornos no plano das fatias) no intervalo que vai da fatia 0 à fatia p e por uma única estrutura (um único contorno no plano das fatias) no intervalo que vai da fatia p à fatia n . Estes dois intervalos definem dois <i>slabs</i> de topologia constante para o objeto 3D.	71
4.2	A figura mostra que um corte ortogonal ao plano das fatias de uma cena 3D. O contorno que representa a borda do objeto na cena 2D resultante do corte ortogonal gera pontos sobre a borda do objeto nas fatias da cena 3D.	71
4.3	(a) Fatia de um joelho obtida por CT na qual dois cortes ortogonais C_1 e C_2 são feitos passando pelo ponto central c marcado pelo usuário no interior do patela (objeto de interesse). (b) Traçado do <i>live-wire</i> sobre a borda do patela no corte ortogonal C_1 . As duas linhas horizontais representam a primeira e a última fatia que contém este osso na cena 3D. (c) Traçado do <i>live-wire</i> sobre a borda do patela no corte ortogonal C_2	74

-
- 4.4 A figura ilustra a seleção de cortes ortogonais para o osso calcâneo em uma cena 3D de MRI. O calcâneo define dois *slabs*, I_1 e I_2 , e três estruturas, S_1 , S_2 e S_3 . S_1 e S_2 pertencem a I_1 e S_3 pertence a I_2 . (a) Uma fatia de I_1 indica as bordas de S_1 e S_2 . Um mesmo plano de corte passando por $c_1 \in S_1$ e $c_2 \in S_2$ é usado para indicar dois cortes ortogonais a S_1 e S_2 respectivamente. As figuras (b) e (c) mostram respectivamente os traçados do *live-wire* para S_1 e S_2 na imagem do corte ortogonal indicado em (a). As linhas horizontais definem as fatias inicial e final de I_1 . (d) Uma fatia de I_2 indica o mesmo plano de corte de (a) passando por $c_3 \in S_3$. (e) Imagem do corte ortogonal mostrando o traçado do *live-wire* para S_3 . As linhas horizontais definem as fatias inicial e final de I_2 75
- 4.5 Os pontos do contorno ortogonal fechado, conectado e orientado levam as informações da orientação da borda e do ângulo de corte quando são selecionados nas fatias. Estas informações são usadas para encontrar a ordem dos pontos na fatia de acordo com a orientação da borda. 77

Lista de Tabelas

3.1	Velocidade de segmentação (fatias/minuto) SP_{ormb} para todos possíveis valores de $o, r, m,$ e b	63
3.2	Diferença em velocidade $ SP_{omb} - SP_{om'b} $ (parte superior) e seu nível de significância estatística (parte inferior) para todos operadores $o,$ objetos b e $m' = MN$ e $m \in \{LW,LL,AL\}$	63
3.3	Velocidade de segmentação SP_{omb} (fatias/minuto) para operador, método e objeto.	64
3.4	Repetibilidade de segmentação $RP_{oor_1r_2mb}$ para todos possíveis valores de $o, r_1, r_2, m,$ e b	65
3.5	Diferença em repetibilidade $ RP_{oomb} - RP_{oom'b} $ (parte superior) e seu nível de significância estatística (parte inferior) para cada operador $o,$ objeto b e $m' = MN$ e $m \in \{LW,LL,AL\}$	66
3.6	Lista de repetibilidade de segmentação $RP_{o_1o_2mb}$ por operadores $(o_1, o_2),$ método (m) e objeto (b)	66
3.7	Lista de repetibilidade de segmentação RP_{mb} por método (m) e objeto (b)	67
4.1	Tempo do usuário (minutos) para segmentar T_a para todos possíveis valores de o e m usando cenas 3D com diferente número de fatias.	79
4.2	Velocidade de segmentação (fatias/minuto) $SP^{(3)}_{oemb}$ para todos possíveis valores de $o, e, m,$ e $b,$ usando uma cena 3D com 32 fatias.	81
4.3	Diferença em velocidade $ SP^{(3)}_{omb} - SP^{(3)}_{om'b} $ (parte superior) e seu nível de significância estatística (parte inferior) para todos operadores $o,$ objetos b e $m' = 3D$ e $m = LL$	81
4.4	Velocidade de segmentação (fatias/minuto) $SP^{(3)}_{omb}$ para todos possíveis valores de $o, m,$ e $b,$ usando uma cena 3D com 32 fatias.	81

4.5	Velocidade de segmentação (fatias/minuto) $SP^{(3)}_{mb}$ para todos possíveis valores de m e b , usando uma cena 3D com 32 fatias.	82
4.6	Repetibilidade $RP^{(3)}_{oo'mb}$ para todos possíveis valores de o , o' , m , e b	83
4.7	Diferença de repetibilidade $ RP^{(3)}_{omb} - RP^{(3)}_{om'b} $ (parte superior) e seu nível de significância estatística (parte inferior) para todos operadores o , objetos b e $m' = 3D$ e $m = LL$	83
4.8	Repetibilidade $RP^{(3)}_{mb}$ para todos possíveis valores de m e b	83

Capítulo 1

Introdução

Em diversas áreas da ciência e engenharia, dados de imagem n-dimensionais são gerados via um dispositivo de aquisição de imagens ou um processo de simulação que captura informações sobre certos fenômenos físicos. Na medicina, especificamente, equipamentos tomográficos produzem diariamente dados de imagem digital 2-, 3- e 4-dimensionais pertencentes a estruturas internas do corpo humano. Neste caso, o fenômeno físico pode ser a forma anatômica ou a função fisiológica de um sistema de estruturas estático ou dinâmico. Os dados de imagem de ressonância magnética de um coração batendo, por exemplo, são dados de imagem 4-dimensionais onde as três primeiras variáveis representam as coordenadas espaciais e a quarta variável representa a coordenada temporal. Portanto, existem certos “objetos” nesses dados (e.g. órgãos do corpo humano) cuja forma e função necessitam ser visualizadas, manipuladas e analisadas para preencher os principais objetivos da digitalização do fenômeno físico [81]. O sucesso da visualização, manipulação e análise de tais objetos depende principalmente de um método exato e eficiente de “extrair” os objetos dos dados de imagem n-dimensionais. Este processo é referido como **segmentação de imagens**.

O enfoque deste trabalho é o problema da segmentação de imagens na área de **computação de imagens médicas** (*three-dimensional medical imaging* [72, 78]). Nesta área, o objetivo principal é permitir que o médico visualize, examine, compreenda, manipule e obtenha medidas de estruturas internas do corpo do paciente de forma não invasiva. Entre seus principais campos de aplicação estão o planejamento de cirurgias, o planejamento de radioterapias e o diagnóstico por imagens. A maioria das aplicações se encontra no campo das cirurgias. Neste campo, as aplicações clínicas mais beneficiadas são o planejamento de cirurgia crânio-maxilo-facial, cirurgia ortopédica, neurocirurgia e cirurgia cardiovascular. A segmentação de imagens é a etapa básica e fundamental em todas estas aplicações.

Métodos de segmentação de imagens médicas podem ser genericamente classificados em dois grupos: **automáticos** e **interativos**. Métodos automáticos [5, 6, 7, 12, 13, 29, 32, 40, 47, 48, 56, 61, 70, 86] evitam intervenção do usuário, mas o sucesso completo destes métodos nem sempre é garantido. Métodos interativos [9, 11, 35, 36, 37, 44, 57, 65, 67, 68, 73, 76, 83] variam de totalmente manuais, onde o usuário pinta as regiões da imagem que correspondem ao objeto (ou traça na imagem as bordas do objeto), a métodos que detectam bordas/regiões de objeto requerendo o mínimo de assistência do usuário. O objetivo deste trabalho é propor um novo paradigma para segmentação interativa formado por três métodos: *live-wire*, *live-lane* e *3D-live-wire*. A escolha entre os métodos propostos vai depender da necessidade de envolvimento do usuário com o processo. Comparados com outros métodos de segmentação, os métodos *live* representam uma solução prática e eficiente que sempre garante o sucesso da segmentação requerendo a assistência do usuário apenas quando necessária.

Para introduzir a relevância dos métodos propostos no contexto de segmentação de imagens, este capítulo é dividido em cinco seções. A seção 1.1 discute a importância da segmentação em computação de imagens médicas e os principais aspectos da segmentação de imagens que motivam este trabalho. Em seguida, o processo de aquisição de dados de imagem e os conceitos elementares associados a estes dados são apresentados na seção 1.2. A seção 1.3 apresenta os métodos *live* e discute o critério de avaliação adotado para validar estes métodos. Na seção 1.4, uma breve comparação é feita entre os métodos propostos e outras técnicas da literatura de segmentação de imagens. A seção 1.5 finaliza o capítulo descrevendo a estrutura do trabalho.

1.1 Motivação

Uma avaliação criteriosa do progresso da computação de imagens médicas nos últimos dez anos [33, 93, 92] revela um rápido crescimento e excelentes perspectivas para o futuro, mas indica a segmentação de imagens como um dos principais problemas no desenvolvimento de aplicações clínicas. O alto grau de incerteza dos objetos em imagens médicas faz com que a presença de um especialista no domínio da aplicação seja crucial ao menos na verificação do resultado da segmentação. O baixo contraste para alguns tipos de tecido e o ruído devido ao processo de aquisição de dados são os fatores que mais influenciam nesta incerteza. Na maioria das aplicações, a segmentação completamente automática não é possível e o especialista precisa interagir manualmente para extrair os objetos da imagem. Este processo pode ser extremamente laborioso e em muitos casos faz com que a aplicação se torne inviável para ser

utilizada na rotina de clínicas e hospitais.

Enquanto os métodos automáticos de segmentação têm sido utilizados em aplicações específicas e de forma dedicada [6, 13, 29, 32, 39, 56, 75, 90], eles falham quando novas situações devido a diferentes protocolos de aquisição de dados, modalidades, ou tipos de objeto são apresentadas. Fazer com que estes métodos funcionem de forma repetitiva para um grande número de dados normalmente requer um esforço considerável em pesquisa e desenvolvimento. Consequentemente, em tais situações, métodos interativos (muitas vezes manuais) são usados. Existem, e continuarão a existir, situações onde os métodos disponíveis falham ou requerem considerável esforço de desenvolvimento e, portanto, a segmentação manual será a única alternativa imediata. Por este ponto de vista, métodos interativos que provêm completo controle ao usuário são mais interessantes, uma vez que eles sempre garantem o sucesso da segmentação, embora possam consumir muito tempo do usuário em algumas situações. Esta observação leva a concluir que duas das principais metas de pesquisa em métodos de segmentação interativa devem ser: (i) prover ao usuário, **tão completo quanto for possível**, o controle sobre o processo de segmentação **enquanto** este estiver em andamento e (ii) minimizar o envolvimento do usuário e o tempo total do usuário requeridos para segmentação. Os métodos *live* são consistentes com estas metas. O aspecto de prover ao usuário completo controle durante a segmentação é fundamental para garantir o resultado por ele desejado no final do processo. A minimização do envolvimento e do tempo total do usuário é crucial para tornar o método viável em situações práticas.

O processo de extrair um objeto de uma imagem pode ser dividido em duas tarefas: **reconhecer o objeto** e **delinear o objeto**. Reconhecer o objeto na imagem consiste em determinar mais ou menos sua localização e distinguir este objeto das outras entidades similares na imagem. Delinear o objeto consiste em definir precisamente a extensão do objeto na imagem. Operadores humanos (especialistas no domínio da aplicação) normalmente são capazes de reconhecer o objeto com maior facilidade do que os algoritmos de computador. A maior razão para esta limitação dos algoritmos de computador é a dificuldade de transformar o conhecimento global relevante sobre o objeto em operações locais computáveis. Isto tem sido um dos maiores obstáculos das técnicas automáticas de segmentação de imagens. Por outro lado, os algoritmos de computador normalmente executam a tarefa de delinear o objeto na imagem bem melhor (mais repetitivo, mais rápido e mais exato) do que os operadores humanos. Uma ação simples de assistência do usuário, como marcar um ponto na imagem, é muitas vezes suficiente para completar o processo de segmentação. Tentar evitar esta intervenção tendo como meta a automação total pode implicar em considerável esforço de pesquisa e desenvolvimento sem produzir nenhum resultado prático no final. Uma estratégia eficaz,

portanto, parece ser explorar ativamente a sinergia entre o operador humano e os algoritmos de computador durante a segmentação. Os métodos *live* utilizam esta estratégia, na qual o objeto de interesse é delineado automaticamente e o usuário interage apenas quando o processo requer uma ação de reconhecimento deste objeto na imagem.

1.2 Conceitos Básicos

Uma imagem digital monocromática [31] é vista como uma matriz $m \times n$, onde os índices de linha e coluna identificam um ponto na imagem e o valor numérico do elemento correspondente na matriz está associado com alguma grandeza física medida neste ponto. Uma imagem tomográfica é uma imagem digital monocromática. A grandeza física depende do dispositivo ou modalidade de aquisição de dados. Na tomografia por raios-X (CT), por exemplo, esta grandeza física é a densidade do material em estudo. Os elementos da matriz são chamados **pixels** representando a abreviação de *picture elements*. Na tela do computador esta matriz é apresentada associando a um ou mais pontos da tela um tom de cinza proporcional ao valor numérico de cada elemento da matriz. Neste trabalho, um **objeto 2D** de interesse na imagem é composto por uma ou mais regiões de pixels conexos. A figura 1.1 apresenta uma imagem tomográfica do joelho de um paciente obtida por CT. O patela (osso da rótula do joelho), indicado na figura 1.1, é um exemplo de objeto 2D formado por uma única região de pixels conexos. Uma imagem tomográfica é portanto um dado de imagem 2-dimensional (2D) que pode conter um ou mais objetos 2D de interesse. Neste trabalho uma imagem tomográfica é referida como uma *cena 2D*.



Figura 1.1: Imagem da fatia de um joelho obtida por CT. A figura indica o patela (osso da rótula do joelho) como exemplo de objeto 2D.

Dados de imagem médica 2-, 3- e 4-dimensionais são normalmente obtidos como imagens tomográficas de cortes (ou fatias) paralelos, consecutivos no espaço (e/ou no tempo) e transversais ao eixo longitudinal do paciente. A figura 1.2 ilustra o processo de aquisição de dados 3-dimensionais (3D). O valor numérico associado a cada pixel das fatias é medido considerando uma pequena região cubóide do espaço. Cada pixel nestas fatias ocupa uma área de dimensão $p \times p$ no espaço. Considerando o espaçamento d entre as fatias, os **voxels** são a extensão 3D dos pixels e representam a abreviação de *volume elements*. Visualizando cada pixel como um voxel no espaço, os dados de imagem 3D formam uma **cena 3D**. Neste trabalho, um **objeto 3D** de interesse em uma cena 3D é composto por uma única região de voxels conexos. Nas imagens das fatias, porém, este objeto pode ser representado por uma ou mais regiões de pixels conexos. A figura 1.3 ilustra como é a projeção na tela do computador de uma cena 3D de um joelho obtida por CT. A figura 1.3a mostra a cena completa e a figura 1.3b mostra a cena com um corte ortogonal ao plano das fatias tornando possível a visualização do interior da cena. Uma **cena 4D** é normalmente adquirida como uma sequência de cenas 3D (ou cenas 2D) de uma mesma região do corpo do paciente em instantes consecutivos de tempo. Neste trabalho, um **objeto 4D** de interesse em uma cena 4D é representado por um mesmo objeto 3D (ou objeto 2D) que pode ter sua posição, e/ou forma, variada no espaço ao longo dos instantes de tempo. Para maiores detalhes sobre o processo de aquisição destas imagens e as operações de processamento de imagens e computação gráfica que podem ser aplicadas a estes dados veja [24, 79, 80].

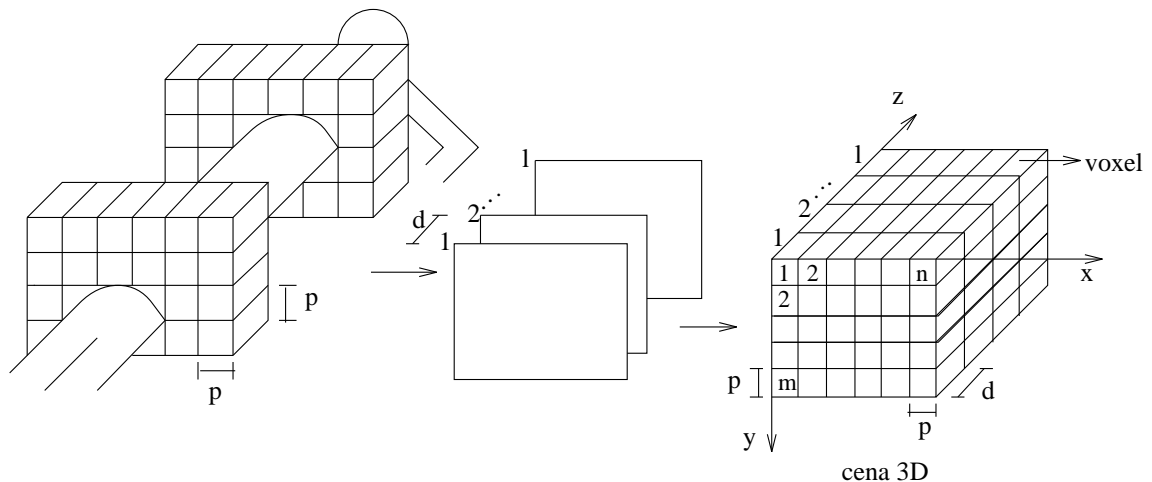


Figura 1.2: Processo de aquisição de imagens tomográficas resultando em uma cena 3D.

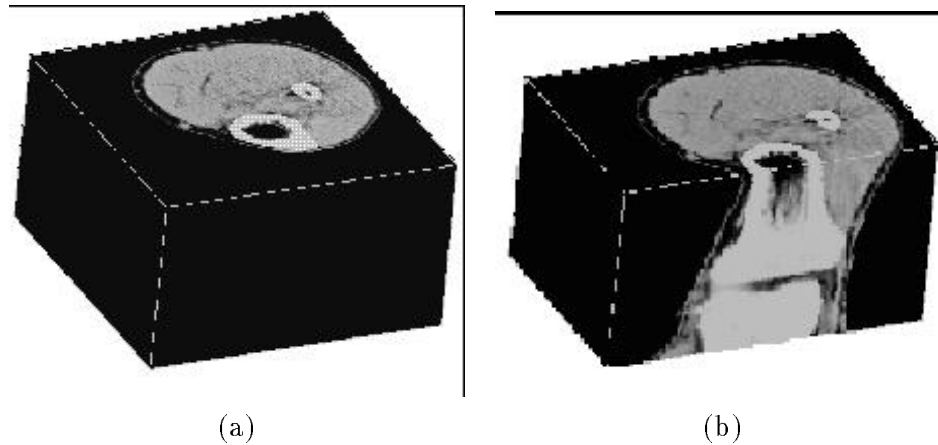


Figura 1.3: As figuras indicam o patela como exemplo de objeto 3D. (a) Cena 3D formada pelo empilhamento das fatias de um joelho. (b) Visualização do interior da cena 3D através de um corte ortogonal ao plano das fatias.

Dada uma cena 2D, 3D, ou 4D, uma borda de objeto pode ser definida pelos elementos da cena (e.g. pixels, arestas de pixels, faces de voxels, etc.) que separam o interior do exterior do objeto e um objeto pode ter várias bordas na cena. Existem diversas estratégias e variações para extrair um objeto de uma cena, de uma forma geral estas estratégias podem ser classificadas em duas abordagens: **técnicas baseadas em região** e **técnicas baseadas em borda**. Técnicas baseadas em região [1, 11, 35, 43, 55] procuram identificar todos os elementos que compõem o interior do objeto incluindo as bordas. Técnicas baseadas em borda [23, 37, 39, 60, 67, 75, 90] identificam apenas os elementos que compõem as bordas do objeto e usam estes elementos para extrair o objeto da cena. Não existe nenhum critério que afirme superioridade de uma destas abordagens sobre a outra. Para uma dada aplicação, uma abordagem pode ser mais apropriada do que a outra, mas o contrário pode ser verificado em uma aplicação diferente. Os métodos *live* são classificados como técnicas baseadas em borda.

Nos métodos *live*, uma borda de objeto 2D, 3D, ou 4D em uma fatia (cena 2D) é definida como um contorno “fechado”, “conectado” e “orientado” formado pelas arestas dos pixels desta fatia (figura 1.4). Estas propriedades são rigorosamente definidas na literatura de espaços digitais [77]. Uma borda fechada é aquela que separa o espaço digital em duas componentes disjuntas de tal forma que qualquer “caminho” que inicie em um pixel de uma das componentes e termine na outra componente “encontra” a borda. Uma borda orientada é aquela em que uma das duas componentes pode ser identificada como constituindo o interior do objeto e a outra representando o exterior. Uma borda conectada é aquela que garante

ser uma única curva conectada que não se cruza em nenhum ponto. Em outras palavras, o contorno que define a borda é uma curva digital de Jordan [34]. Portanto, um objeto 2D, 3D, ou 4D, pode ser representado por um ou mais contornos fechados, conectados e orientados por fatia. Uma forma de segmentar este objeto é delinear todas suas bordas fatia por fatia.

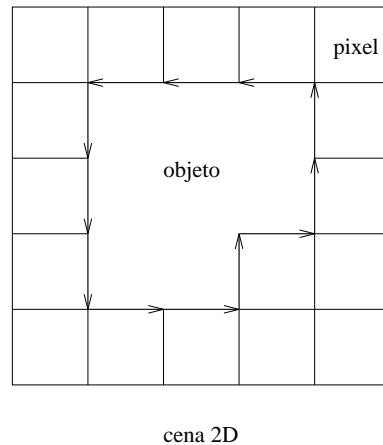


Figura 1.4: Borda de um objeto em uma cena 2D definida como um contorno fechado, conectado e orientado formado por arestas de pixel.

1.3 Métodos Propostos

Considerando que o usuário médico está acostumado a reconhecer os objetos nas imagens das fatias e que a interface que os usuários normalmente dispõem é uma interface 2D representada pela tela do computador, a segmentação de cenas 2D, 3D e 4D fatia por fatia é a forma mais natural de prover interação e controle do usuário sobre o processo ¹. Nos métodos *live-wire* e *live-lane* [26, 27], o usuário interage com o programa para realizar a segmentação fatia por fatia. No método *3D-live-wire* [25], o usuário tem uma rápida participação inicial e a segmentação é feita automaticamente fatia por fatia. O usuário acompanha este processo e pode intervir na segmentação de uma fatia se for necessário. No caso de intervenção, o usuário usa o *live-wire* para assistir na segmentação desta fatia. Assim os métodos *live* sempre garantem o sucesso da segmentação.

Nos métodos *live*, para extrair uma borda de objeto em uma cena 2D, o problema de encontrar um segmento ótimo de borda na cena é transformado no problema de encontrar o caminho de custo mínimo entre dois vértices de um grafo. A estratégia utilizada pelos métodos

¹Com os avanços da realidade virtual é possível que no futuro técnicas interativas possam usar uma interface 3D para prover controle mais eficiente ao usuário sobre o processo de segmentação.

divide a segmentação em duas etapas: **classificação** e **conectividade**. A classificação é a etapa que associa um valor de custo para cada arco do grafo e a conectividade é a etapa que encontra o caminho de custo mínimo entre dois vértices do grafo. Para tanto, uma cena 2D é vista como um grafo direcionado onde os vértices dos pixels representam os vértices do grafo e as arestas orientadas dos pixels representam os arcos. Cada aresta pode ter duas possíveis orientações definindo portanto dois arcos no grafo. Na classificação, um conjunto de características de imagem (gradiente, intensidade de pixel, etc.) é associado para cada aresta orientada de pixel. Os valores destas características são convertidos em um único valor de custo no intervalo $[0,1]$ por aresta orientada. A classificação associa custos baixos para as arestas orientadas que têm propriedades similares à borda orientada (a orientação da borda é definida no início do processo) e custos altos às demais. Na etapa de conectividade, para quaisquer dois pontos (vértices de pixel) especificados na borda, o problema de encontrar o melhor segmento de borda (como um conjunto de arestas de pixel orientadas e conectadas) entre os dois pontos é transformado no problema de encontrar o caminho de custo mínimo entre os dois vértices dados no grafo. Este problema é resolvido por programação dinâmica [4] usando algoritmos de busca em grafo [17, 53].

1.3.1 *Live-Wire, Live-Lane e 3D-Live-Wire*

No método *live-wire*, para segmentar uma borda de objeto em uma cena 2D, o usuário especifica um ponto inicial sobre a borda usando o cursor do *mouse*. Para qualquer posição subsequente do cursor na cena, dado que o usuário pode mover o *mouse* posicionando o cursor em qualquer lugar da cena, uma curva - caminho ótimo (custo mínimo) - conectando o ponto inicial ao ponto que corresponde à posição atual do cursor é calculada e apresentada na tela do computador em tempo real, portanto o nome “live-wire”. Quando o cursor é posicionado próximo a esta borda, o segmento de *live-wire* adere à borda. Se este segmento descrever adequadamente a borda desejada, o usuário deposita o cursor cuja localização passa a ser a do novo ponto de partida. A borda completa na cena 2D (contorno fechado, conectado e orientado) é especificada via um conjunto de segmentos de *live-wire* desta maneira. Este processo é repetido para cada borda de objeto fatia por fatia. Normalmente de 2 a 5 pontos são necessários completar uma borda de objeto em uma fatia. Note que a tarefa de reconhecer o objeto é feita pelo usuário no momento que marca um ponto sobre a borda, enquanto que a tarefa de delinear o segmento ótimo de borda é feita pelo computador.

No método *live-lane*, o envolvimento do usuário é mais ativo e mais fortemente integrado com as ações do computador. Para extrair uma borda de objeto em uma cena 2D,

o usuário seleciona um ponto inicial e subsequentemente conduz o cursor na vizinhança da borda dentro de uma faixa de certa largura, portanto o nome “live lane”. Durante o desenho do percurso conduzido pelo usuário, pontos são selecionados de forma intermitente e automática pelo computador e segmentos de *live-wire* entre sucessivos pontos são calculados e apresentados na tela do computador em tempo real. A largura da faixa pode variar ao longo da borda, diminuindo em regiões onde a borda é mal definida e aumentando onde existe forte evidência da borda do objeto. Novamente, o usuário é o melhor juiz para reconhecer o tipo de borda apresentado em diferentes partes da cena (tarefa de reconhecer o objeto) e o computador “ler” suas ações para inferir o tipo de borda durante a tarefa de delinear o objeto. Como o usuário tende a conduzir o cursor mais rápido nas partes bem definidas da borda, esta leitura é feita medindo-se a velocidade e a aceleração dos movimentos do cursor durante o percurso. Quando o movimento é rápido, a largura da faixa aumenta e o método se aproxima do método *live-wire*. Quando o movimento é lento, o método torna-se quase um traçado manual. Assim *live-lane* é um processo elegante que opera em um contínuo entre *live-wire* e o traçado manual.

No método *3D-live-wire*, dada uma cena 3D/4D, a segmentação é feita visando extrair cada objeto 3D de cada cena 3D separadamente e explorando a seguinte propriedade: qualquer secção de corte de uma cena 3D resulta em uma subcena 2D na qual uma borda de objeto é representada por um contorno fechado, conectado e orientado. Nesta subcena 2D, o objeto 3D pode ter várias bordas. As fatias de uma cena 3D são um caso particular desta propriedade. A figura 1.3b, por exemplo, mostra que uma secção de corte ortogonal ao plano das fatias define uma subcena 2D no interior da cena 3D da figura 1.3a. Portanto, para extrair um objeto 3D de uma cena 3D, o usuário especifica inicialmente contornos fechados, conectados e orientados usando *live-wire* para delinear as bordas do objeto em algumas imagens de cortes ortogonais ao plano das fatias (normalmente de 1 a 6 cortes). Se o usuário selecionar estes cortes de forma estratégica, um número suficiente de pontos é gerado sobre estas bordas em todas as fatias da cena 3D. Em seguida, para cada fatia, estes pontos são separados e associados à borda correspondente na fatia. O conjunto de pontos associados a uma borda de objeto em uma dada fatia tem uma ordem de processamento tal que, a borda completa na fatia é encontrada calculando e apresentando em tempo real cada segmento de *live-wire* entre cada dois pontos consecutivos. Este processo é repetido automaticamente para todas as bordas do objeto fatia por fatia.

Nos métodos *live-wire* e *live-lane*, o usuário identifica e traça cada borda do objeto 3D em uma dada fatia isoladamente. No método *3D-live-wire*, o problema de separar os pontos que pertencem a bordas diferentes do objeto 3D em uma dada fatia é resolvido subdividindo

a cena 3D em intervalos de fatias consecutivas, onde o objeto é representado por um número constante de contornos por fatia. Em cada intervalo, o objeto 3D é composto por uma ou mais estruturas 3D (regiões de voxels conexos). Considerando a subcena 3D formada pelas fatias de um dado intervalo, as estruturas 3D desta subcena são desconexas entre si. O usuário especifica contornos em cortes ortogonais para cada estrutura separadamente e a tarefa de delinear múltiplos contornos por fatia se torna um processo automático e robusto. Obviamente a performance do algoritmo está comprometida com o número de fatias da cena 3D *versus* o número de cortes necessários. O número de cortes depende do tipo de borda, da quantidade de vezes que a topologia do objeto varia ao longo das fatias e da qualidade do processo de classificação. Por isto, em algumas situações, o paradigma da segmentação guiada pelo usuário pode favorecer métodos como *live-wire* e *live-lane*. A principal vantagem do *3D-live-wire* é que quando poucos cortes são suficientes para delinear as bordas do objeto nas fatias, o usuário executa o método *live-wire* em um número de cortes ortogonais bem menor do que o número de fatias reduzindo bastante o tempo do usuário no processo de segmentação.

1.3.2 Metodologia de Avaliação

Na avaliação de métodos de segmentação guiada pelo usuário, três fatores parecem ter maior importância: “velocidade”, “repetibilidade” e “exatidão”. Velocidade expressa a extensão do envolvimento do usuário no processo. Repetibilidade expressa as variações no resultado da segmentação como um efeito das ações tomadas pelo usuário durante o processo. Exatidão expressa o quanto o resultado da segmentação está próximo da verdade. A utilidade de um método como função destes três fatores também pode considerar diferentes ênfases para cada um deles dependendo da aplicação. Por exemplo, definir “verdade” no caso de imagens médicas é uma tarefa muito difícil. Testar a exatidão de um método usando como “verdade” modelos (*phantoms*) físicos ou matemáticos nunca vai refletir a situação real. Como o próprio processo de aquisição de dados é incerto em reproduzir o objeto verdadeiro, a exatidão da segmentação fica sob julgamento do usuário médico com relação a utilidade do resultado. Neste trabalho, portanto, os métodos são avaliados sob supervisão de um médico especialista no domínio da aplicação e em função da velocidade e repetibilidade.

A avaliação dos métodos *live* é realizada em duas etapas. Na primeira etapa, os métodos *live-wire* e *live-lane* são comparados com a segmentação manual. Na segunda etapa, o método *3D-live-wire* é comparado com o *live-lane*. Conclui-se que *live-wire* e *live-lane* são estatisticamente significativamente mais repetitivos e 1,5-2,5 vezes mais rápidos do que o

traçado manual. O *3D-live-wire* é estatisticamente significativamente mais repetitivo que o *live-lane*. O tempo do usuário para segmentação usando *3D-live-wire* depende apenas do número de cortes necessários para segmentar o objeto. Portanto, para um número fixo de cortes, sua velocidade de segmentação (fatias/minuto) varia com o número de fatias. No caso de uma cena 3D com 63 fatias em que o objeto de interesse requer 5 cortes para ser segmentado, o *3D-live-wire* é 6 vezes mais rápido que o *live-lane* e conseqüentemente 15 vezes mais rápido que o traçado manual. Nesta cena o usuário gasta cerca de 2 minutos usando o *3D-live-wire*.

1.4 Comparação com Outros Métodos

A idéia de transformar o problema de detecção de borda em um problema de busca em grafo foi inicialmente proposta por Martelli [46] na qual o algoritmo A^* [53] é utilizado. Para detectar um segmento de borda, que parte de um pixel x_a e chega a um pixel x_b selecionados sobre a borda do objeto na imagem, este segmento é representado pelo caminho “ótimo” que parte de um nó n_a (pixel x_a) e chega a um nó n_b (pixel x_b) em um grafo G direcionado. No algoritmo A^* , uma função de custo $f(x_i) = \tilde{g}(x_i) + \tilde{h}(x_i)$ é definida para representar o custo de um caminho entre x_a e x_b passando por um terceiro pixel x_i . O termo $\tilde{g}(x_i)$ é uma estimativa do custo de um caminho que parte de n_a e chega ao nó n_i (pixel x_i) e o termo $\tilde{h}(x_i)$ é uma estimativa do custo de um caminho que parte de n_i e chega a n_b . Qualquer informação heurística usada para resolver este problema é representada pelo termo $\tilde{h}(x_i)$. Quando $\tilde{h}(x_i) = 0$ (nenhuma heurística), o algoritmo A^* coincide com o algoritmo de Dijkstra [20], o qual sempre garante o caminho ótimo como o caminho de custo mínimo global em G [53]. Quando $\tilde{h}(x_i) \neq 0$, o algoritmo A^* nem sempre garante o caminho ótimo mas pode encontrar a solução do problema mais rápido que o algoritmo de Dijkstra. Na tentativa de reduzir o tempo de processamento do algoritmo A^* , variações que usam diferentes heurísticas são discutidas em [3, 66]. O grafo utilizado nestes trabalhos, porém, nem sempre garante que existe pelo menos um caminho que parte de x_a e chega a x_b . Nos métodos *live*, x_a e x_b são vértices de pixel e o grafo G utilizado sempre garante que para qualquer nó inicial de G existe pelo menos um caminho que parte deste nó e chega a qualquer outro nó de G . Neste trabalho são propostos um algoritmo de Dijkstra modificado e algumas variações deste algoritmo, a solução do problema de detecção de um segmento de borda é obtida em tempo real e o mecanismo de interação com o usuário sempre garante o sucesso da segmentação.

O algoritmo de busca em grafo dos métodos *live* pode ser visto como um algoritmo

de programação dinâmica [4, 46]. Programação dinâmica tem sido utilizada no passado para resolver o problema de detecção de borda impondo restrições ao formato da borda e fazendo com que o algoritmo seja limitado a algumas aplicações [22, 50, 58]. Nos métodos *live*, uma borda de objeto em uma cena 2D é um contorno fechado, conectado e orientado, mas não existe nenhuma restrição para o formato deste contorno. Estas restrições são completamente eliminadas por elaborar o problema usando o grafo direcionado cujos arcos são as arestas orientadas dos pixel da cena. As propriedades geométricas de tal grafo permitem que os algoritmos padrões de programação dinâmica sejam modificados para encontrar a resposta em tempo real requerida para algoritmos interativos de segmentação de imagens.

A dificuldade básica da segmentação de imagens está na integração de características locais da imagem (e.g. intensidade de pixel, gradiente, curvatura) com a informação global sobre o objeto de interesse (e.g. localização, formato, topologia). Uma abordagem para resolver este problema é o uso de uma linguagem intermediária, a qual atua sobre as características locais mas pode ser influenciada pelas informações globais. O método de contorno ativo (ou *snake*) inicialmente proposto por Kass *et al.* [37] é uma forma de superar esta dificuldade. A idéia básica é a utilização de um modelo de curva paramétrica que, após ser posicionado na imagem, é iterativamente deformado por ação de forças internas e externas aplicadas aos pontos de controle da curva visando minimizar um funcional de energia. Esta energia é dividida em uma componente interna associada aos parâmetros de deformação da curva (rigidez e elasticidade) e uma componente externa definida pelas características locais da imagem (e.g. pixels com alto valor de gradiente evidenciam a presença de borda e portanto correspondem a pontos de baixa energia). Métodos de contornos ativos têm sido bastante populares na literatura de segmentação de imagens [10, 18, 41, 42, 44, 52, 88]. A idéia tem sido estendida para objetos 3D (“balões” [14, 15]) e, como consequência, outros métodos baseados em contornos deformáveis têm sido propostos [16, 54, 69, 71] visando outros mecanismos de minimização de funcionais de custo ou energia. O posicionamento inicial do contorno na imagem é normalmente feito pelo usuário que em seguida espera que o contorno se deforme iterativamente até que, no valor de ótimo da função utilizada, o contorno coincida com a borda desejada. É fácil imaginar que devido a possível existência de mínimos locais na imagem, este valor ótimo pode não coincidir com a borda desejada. Nestas situações, o usuário tem que reiniciar o processo ou corrigir a borda obtida manualmente. O usuário pode também interferir durante a segmentação alterando a posição dos pontos de controle do contorno na tentativa de acelerar o processo de convergência [9]. Entretanto, controlar o comportamento do modelo não parece ser uma tarefa fácil visto que ações locais do usuário podem afetar o formato global do contorno. Além dos problemas de interação com o usuário, modelos de contornos ativos

(ou deformáveis) normalmente apresentam problemas de estabilidade, dificuldade de aderir a protusões e identações da borda, dificuldade de parar quando uma descontinuidade da borda é encontrada, etc. Uma questão fica em aberto quanto a eficiência destes métodos em prover completo controle ao usuário durante o processo e quanto a reduzir o tempo total do usuário para segmentação. Por outro lado, a abordagem utilizada nos métodos *live* é rápida, repetitiva, exata, provê controle eficiente ao usuário e sempre garante a borda desejada no final do processo.

O método *live-wire* tem sido paralelamente pesquisado por Mortensen e Barrett [51] e utilizado para composição de imagens. Nesta aplicação, o *live-wire* é chamado de *intelligent scissors* e difere do *live-wire* proposto nos métodos *live* em muitos aspectos: as características de imagem utilizadas, o mecanismo de classificação, o grafo utilizado e o tipo de elemento de borda. O elemento de borda utilizado é o pixel e um arco do grafo conecta dois pixels de vizinhança-8. O algoritmo de Dijkstra é usado e mostra-se que o *live-wire* é mais rápido e mais exato que o traçado manual.

1.5 Estrutura do Trabalho

A diferença básica entre os métodos *live* está no mecanismo de interação com o usuário. Esta diferença pode ser analisada considerando o espaço de segmentação proposto na figura 1.5. Em um extremo da figura, as tarefas de reconhecer e delinear o objeto são realizadas manualmente (e.g. traçado manual) enquanto no outro extremo ambas tarefas são realizadas sem intervenção do usuário (e.g. métodos totalmente automáticos tais como *thresholding* [31]). Nos métodos *live-wire* e *live-lane*, a tarefa de reconhecer a borda é realizada manualmente pelo usuário e no método *3D-live-wire* esta tarefa é semi-automática. O *live-lane* requer maior envolvimento do usuário para delinear a borda do que o *live-wire* e no *3D-live-wire* a borda é delineada automaticamente. Em todos os métodos o usuário tem completo controle sobre a segmentação, podendo interromper e interagir em qualquer etapa do processo.

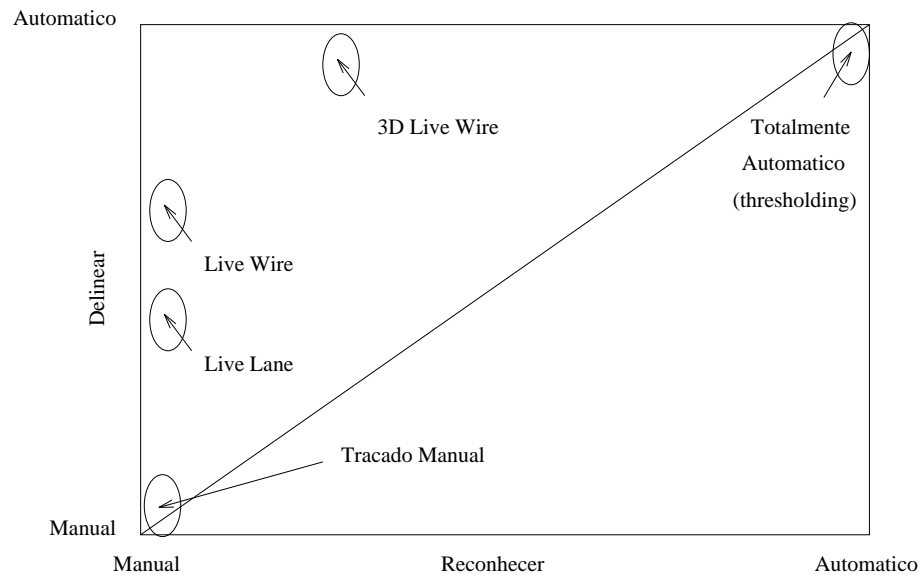


Figura 1.5: Espaço de Segmentação.

A estratégia de dividir o processo de segmentação nas etapas de classificação e conectividade é a parte comum aos métodos *live-wire*, *live-lane* e *3D-live-wire*. Esta estratégia é apresentada no capítulo 2. Os métodos *live-wire* e *live-lane* requerem interação fatia por fatia e, portanto, são apresentados e avaliados juntos no capítulo 3. O método *3D-live-wire* se destaca dos outros por explorar a informação tridimensional inerente aos dados tomográficos reduzindo o envolvimento e o tempo total do usuário. Este método é apresentado e avaliado no capítulo 4. O capítulo 5 conclui os resultados deste trabalho e discute trabalhos futuros que podem se beneficiar da mesma estratégia de separar a segmentação em classificação e conectividade explorando a sinergia entre operador humano e algoritmos de computador.

Capítulo 2

Classificação e Conectividade

Na literatura de processamento e análise de imagens [31, 45, 63], segmentar uma imagem é dividi-la em regiões (ou segmentos) que guardam certas propriedades relevantes para identificar, extrair, descrever e analisar diferentes objetos na imagem. Em reconhecimento de padrões [21, 28, 64], estas regiões resultam de um processo que classifica os pixels da imagem em diferentes classes (e.g. no caso de um único objeto, as classes são: objeto e não objeto). Quando esta classificação falha e, além dos segmentos que descrevem um objeto de interesse, segmentos de outros objetos e dejetos de segmentos são gerados, a segmentação requer uma classificação complementar. Nesta segunda etapa, operações baseadas na geometria dos segmentos (e.g. conectividade, formato e tamanho dos segmentos) são aplicadas sobre a imagem parcialmente segmentada para completar a definição do objeto. Um exemplo desta situação é ilustrado na figura 2.1 na qual o patela é usado como objeto de interesse de uma cena 3D. A figura 2.1a mostra a reconstrução 3D do resultado de uma segmentação onde os voxels com intensidade acima de um dado valor limiar são classificados como patela. Obviamente esta classificação é insuficiente para isolar este osso dos demais ossos do joelho. Entretanto, note que o patela é representado por um único segmento de voxels desconexo dos demais segmentos. Um ponto semente selecionado no seu interior é suficiente para reclassificar como patela apenas os voxels conexos a este ponto, isolando o patela dos demais ossos como ilustrado na figura 2.1b. Métodos de segmentação que possuem estas duas etapas têm maiores chances de sucesso. Nos métodos *live*, estas duas etapas são chamadas **classificação e conectividade**.

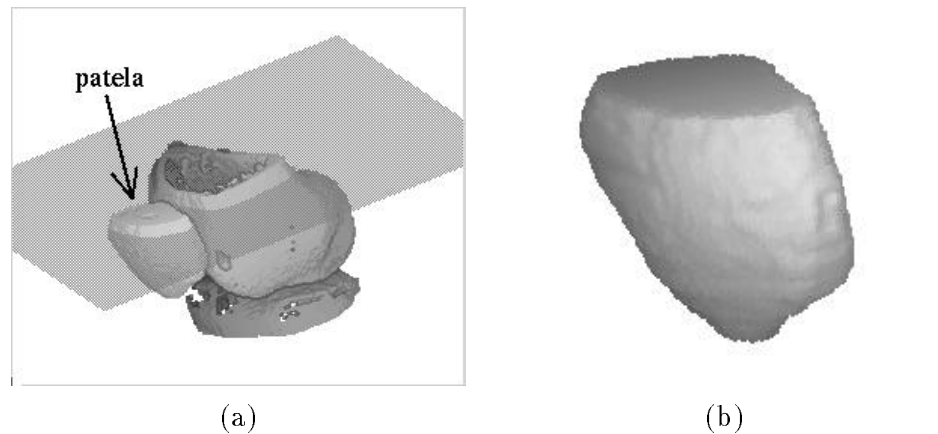


Figura 2.1: A figura mostra a reconstrução 3D dos ossos do joelho após segmentação. (a) Segmentação usando apenas classificação por limiar. Insuficiente para separar o patela dos outros ossos. (b) Segmentação usando classificação por limiar seguida de conectividade. Apenas o patela é extraído da cena 3D.

A classificação e a conectividade envolvem conceitos binário e *fuzzy*. Definindo uma cena como o conjunto universo cujos elementos (pixels ou voxels) podem pertencer ou não a um objeto de interesse contido nesta cena, a classificação binária associa a cada elemento da cena um valor de pertinência no conjunto $\{0,1\}$ enquanto a classificação fuzzy [74] associa valores de pertinência no intervalo $[0,1]$. A conectividade é aplicada a uma cena previamente classificada para reclassificar apenas os elementos conexos a elementos sementes selecionados no interior/borda do objeto de interesse. Na conectividade, os elementos da cena podem ser classificados como conexos ou desconexos (conectividade binária) ou em uma escala contínua de 0 a 1 de força de conectividade [84]. Neste último caso, o objeto resultante é uma entidade fuzzy que pode ser tratada dentro da lógica apresentada em [38]. Os métodos *live* adotam uma classificação fuzzy seguida de uma conectividade binária onde o elemento da cena utilizado é a aresta de pixel.

Este capítulo apresenta a estratégia de classificação e conectividade usada pelos métodos *live-wire*, *live-lane* e *3D-live-wire* discutindo possíveis variações que podem ser úteis em algumas aplicações. A seção 2.1 descreve a etapa de classificação e a seção 2.2 apresenta e discute as vantagens e desvantagens de dois algoritmos de conectividade. A seção 2.3 resume os principais aspectos da estratégia de classificação *versus* conectividade, apresenta um exemplo de classificação e conectividade e conclui o capítulo mostrando como esta estratégia é utilizada nos métodos *live* para obter um contorno fechado, conectado e orientado.

2.1 Classificação

Uma cena 2D, denotada \mathbf{C} , é um par (C, g) que consiste de uma matriz C de pixels, finita e retangular, chamada **domínio da cena** e de uma função $g(p) : C \rightarrow [L, H]$ chamada **intensidade da cena** que associa a cada pixel p em C um valor de intensidade dentro do intervalo $[L, H]$. Uma borda de objeto em \mathbf{C} é um contorno fechado, conectado e orientado formado por arestas de pixel conforme ilustrado na figura 1.4. Cada aresta de pixel em \mathbf{C} tem associada duas possíveis orientações. Toda aresta de pixel orientada em \mathbf{C} é em potencial um elemento de borda chamado *bel* (abreviação de *boundary element*). Um bel b de \mathbf{C} é portanto um par ordenado (p, q) de pixels 4-adjacentes (pixels que compartilham uma aresta). Todo bel $b = (p, q)$ de \mathbf{C} tem uma localização e uma orientação. A localização de b é determinada como sendo a única aresta comum a p e q . A orientação de b é assumida tal que p está sempre à esquerda de b e q está sempre à direita (ver figura 2.2).

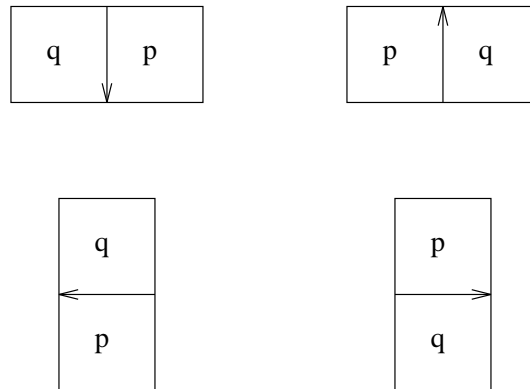


Figura 2.2: Um elemento da borda (bel) é uma aresta de pixel orientada. Os quatro possíveis tipos de bels em uma cena 2D são mostrados. Para qualquer orientação da borda, um bel $b = (p, q)$ é sempre definido tal que p está à esquerda de b e q está à direita. Para uma borda orientada no sentido anti-horário, o pixel p está no interior do objeto e o pixel q no exterior.

A etapa de classificação tem por objetivo associar a cada bel b em \mathbf{C} um valor no intervalo $[0, 1]$ que representa o custo de ter b como elemento da borda desejada. Uma “boa” classificação deve associar custos baixos aos bels da borda desejada e custos altos aos demais. Este processo associa inicialmente um conjunto de características de imagem a todo bel b de \mathbf{C} . Em seguida, estas características são convertidas em um valor de custo conjunto para cada b . Para descrever este processo completamente, as seguintes questões precisam ser abordadas: como é feita a seleção de características de imagem (seção 2.1.1), como converter valores de característica em valores de custo (seção 2.1.2) e como definir os parâmetros destas duas

operações para que custos baixos sejam associados aos bels da borda desejada e custos altos aos demais (seção 2.1.3).

2.1.1 Seleção de Características de Imagem

Uma característica de imagem é uma função que associa para todo bel $b = (p, q)$ de uma dada cena $\mathbf{C} = (C, g)$ um número inteiro representando um valor de propriedade. As características procuram descrever propriedades do “interior” do objeto, do “exterior” do objeto e da própria borda. Existem várias funções que podem ser usadas como característica para classificação de bels. Estas funções podem ser divididas em dois grupos: *características independentes da orientação da borda* e *características sensíveis à orientação da borda*. Existem duas possíveis orientações para delinear uma borda de objeto em \mathbf{C} : sentido horário e sentido anti-horário. Para uma borda no sentido anti-horário, por exemplo, os pixels p dos bels $b = (p, q)$ que formam esta borda estão no interior do objeto e os pixels q estão no exterior (figuras 1.4 e 2.2). Isto significa que para um dado bel b de \mathbf{C} , candidato a elemento da borda desejada, o interior do objeto é suposto como estando à esquerda de b e o exterior à direita. Se uma função leva em conta esta suposição e associa valores diferentes de propriedade para bels de orientações opostas que compartilham o mesmo par de pixels, a característica é dita ser sensível à orientação da borda. A idéia serve para distinguir segmentos de borda com contrastes similares e orientações opostas. Por outro lado, características independentes da orientação da borda associam valores iguais para bels que compartilham o mesmo par de pixels.

As características de imagem implementadas no decorrer deste trabalho são descritas a seguir.

1. Características Independentes da Orientação.

(a) Intensidade máxima em $b = (p, q)$ (f_1):

$$f_1(b) = \begin{cases} g(p), & \text{se } g(p) \geq g(q) \\ g(q), & \text{no caso contrário.} \end{cases} \quad (2.1)$$

(b) Intensidade mínima em $b = (p, q)$ (f_2):

$$f_2(b) = \begin{cases} g(p), & \text{se } g(p) \leq g(q) \\ g(q), & \text{no caso contrário.} \end{cases} \quad (2.2)$$

(c) **Magnitudes de Gradiente** (f_3, f_4, f_5, f_6):

Estas características representam magnitudes de gradiente de g (intensidade da cena) calculadas no centro de b usando as seguintes aproximações digitais (ver figura 2.3).

$$f_3 = (|g(p) - g(q)|) \quad (2.3)$$

$$f_4 = \frac{1}{3}(|g(p) + g(t) + g(v) - g(q) - g(u) - g(w)|) \quad (2.4)$$

$$f_5 = \frac{1}{2}(|g(p) + \frac{1}{2}g(t) + \frac{1}{2}g(v) - g(q) - \frac{1}{2}g(u) - \frac{1}{2}g(w)|) \quad (2.5)$$

$$f_6 = \frac{1}{4}(|g(p) - g(u)| + |g(t) - g(q)| + |g(p) - g(w)| + |g(v) - g(q)|). \quad (2.6)$$

t	u
p ^b	q
v	w

Figura 2.3: Vizinhança de b usada para definir f_3, f_4, f_5 , e f_6

Estas características são claramente não independentes. Na prática porém algumas delas têm desempenho melhor que outras para diferentes aplicações.

2. Características Sensíveis à Orientação.

(a) **Intensidade à direita de $b = (p, q)$** (f_7):

$$f_7(b) = g(q). \quad (2.7)$$

Note que independente da orientação da borda, q é sempre o pixel à direita de $b = (p, q)$ e p é sempre o pixel à esquerda (figura 2.2). Para uma borda no sentido anti-horário, $g(q)$ passa a ser uma estimativa do valor de intensidade no exterior do objeto.

(b) **Intensidade à esquerda de $b = (p, q)$** (f_8):

$$f_8(b) = g(p). \quad (2.8)$$

(c) **Magnitudes de gradiente sensíveis à orientação** ($f_9, f_{10}, f_{11}, f_{12}$):

Existem várias maneiras de estimar um vetor gradiente no centro de $b = (p, q)$. O sinal deste vetor é considerado positivo, sem perda de generalidade, se ele apontar para o lado esquerdo de b (pixel p), negativo se apontar para o lado direito de b (pixel q) e nulo se a direção do vetor coincidir com a direção da aresta de pixel representada por b . A magnitude deste vetor sinalizada pela regra acima é um exemplo de magnitude de gradiente sensível à orientação. Neste trabalho, estas características são definidas combinando o sinal do vetor gradiente da característica f_9 com outras funções de magnitude de gradiente.

$$f_9(b) = g(p) - g(q) \quad (2.9)$$

$$f_{10}(b) = \text{sinal}(f_9(b)) * f_4(b) \quad (2.10)$$

$$f_{11}(b) = \text{sinal}(f_9(b)) * f_5(b) \quad (2.11)$$

$$f_{12}(b) = \text{sinal}(f_9(b)) * f_6(b), \quad (2.12)$$

onde

$$\text{sinal}(f_9(b)) = \begin{cases} +1, & \text{se } g(p) > g(q) \\ -1, & \text{se } g(p) < g(q) \\ 0, & \text{caso contrário.} \end{cases}$$

(d) **Intensidades máxima e mínima sensíveis à orientação** (f_{13}, f_{14})

Em geral, usando o sinal de um vetor gradiente qualquer estimado em b , qualquer característica independente da orientação da borda pode ser transformada em característica sensível à orientação. As características $f_9, f_{10}, f_{11}, f_{12}$ e as características listadas abaixo são exemplos desta regra.

$$f_{13}(b) = \text{sinal}(f_9(b)) * f_1(b) \quad (2.13)$$

$$f_{14}(b) = \text{sinal}(f_9(b)) * f_2(b). \quad (2.14)$$

2.1.2 Geração de Custos

O custo de um bel b de \mathbf{C} ser considerado um elemento da borda desejada é inversamente proporcional ao valor de pertinência de b com relação a esta borda em \mathbf{C} . Portanto,

custos podem ser gerados a partir de qualquer método probabilístico ou fuzzy de gerar funções de pertinência. Neste trabalho, os valores das características $f_i(b)$, $1 \leq i \leq 14$, associados com cada bel b de \mathbf{C} são inicialmente convertidos para valores de custo $c_j(f_i(b))$ no intervalo $[0, 1]$ usando funções c_j referidas como **transformadas de característica**. Estas transformadas são listadas a seguir.

1. **Linear** (c_1):

c_1 é um mapeamento linear dentro do intervalo $[l_1, h_1]$ de valores de característica. Valores fora deste intervalo são mapeados para 1.

$$c_1(x) = \begin{cases} 1, & x \leq l_1 \\ \frac{-x+\mu_1}{\mu_1-l_1}, & l_1 \leq x \leq \mu_1 \\ \frac{x-\mu_1}{h_1-\mu_1}, & \mu_1 \leq x \leq h_1 \\ 1, & x \geq h_1, \end{cases} \quad (2.15)$$

onde μ_1 é um parâmetro livre dentro do intervalo $[l_1, h_1]$.

2. **Gaussiana** (c_2):

c_2 é o complementar de uma função Gaussiana com média μ_2 e desvio padrão σ_2 dentro do intervalo $[l_2, h_2]$. Valores fora deste intervalo são mapeados para 1.

$$c_2(x) = \begin{cases} 1, & x \leq l_2 \\ 1 - \exp \frac{-(x-\mu_2)^2}{2\sigma_2^2}, & l_2 < x < h_2 \\ 1, & x \geq h_2, \end{cases} \quad (2.16)$$

onde $\mu_2 \in [l_2, h_2]$.

3. **Parabólica** (c_3):

c_3 é uma função parabólica no intervalo $[l_3, h_3]$ e os valores fora deste intervalo são mapeados para 1.

$$c_3(x) = \begin{cases} 1, & x \leq l_3 \\ \left(\frac{x-l_3}{h_3-l_3}\right)^2, & l_3 < x < h_3 \\ 1, & x \geq h_3, \end{cases} \quad (2.17)$$

onde $\mu_3 \in [l_3, h_3]$.

O custo conjunto $c(b)$ associado a cada bel b de \mathbf{C} é uma combinação linear dos custos resultantes de cada característica:

$$c(b) = \frac{\sum_i w_i c_{f_i}(f_i(b))}{\sum_i w_i}, \quad (2.18)$$

onde w_i é uma constante positiva indicando a ênfase dada a característica f_i e c_{f_i} é o custo associado com a característica f_i .

Note que qualquer uma das transformadas pode ser usada com qualquer uma das características descritas na seção 2.1.1. O custo conjunto é o resultado da combinação de características e transformadas. A combinação mais apropriada depende da aplicação. Nem todas as características são relevantes para uma dada aplicação. Deve ainda existir outras características e transformadas não consideradas neste trabalho que são mais apropriadas para uma dada aplicação. Os parâmetros das transformadas devem ser escolhidos com a intenção de minimizar o custo conjunto dos bels que pertencem à borda desejada. Por exemplo, se a borda do objeto possui altos valores da característica f_6 , uma combinação razoável é usar a transformada c_3 com $\mu_3 = h_3 =$ ao valor máximo de f_6 em \mathbf{C} e $l_3 = 0$. Em particular, para as transformadas acima, a situação ideal é encontrada quando o ponto de mínimo da curva da transformada coincide com o valor da característica que ocorre com maior frequência na borda do objeto (a moda estatística).

Dada uma aplicação qualquer, o usuário tem completo controle para selecionar características, selecionar qual transformada vai com qual característica, escolher os parâmetros das transformadas e os pesos da equação 2.18. Para facilitar o trabalho do usuário, que na maioria das vezes não é um especialista em processamento de imagens e pode não ter interesse em entender o mecanismo da classificação, uma etapa de treinamento pode ser realizada antecedendo a seleção de características e transformadas conforme descrito na próxima seção.

2.1.3 Treinamento

Os objetivos principais do treinamento são definir uma única orientação para a borda de interesse e guiar o usuário no processo de seleção de características e determinação de parâmetros das transformadas. Os parâmetros das transformadas são calculados baseados na estimativa de parâmetros estatísticos das densidades de probabilidade das características para a borda desejada. Este processo é referido como **seleção automática de parâmetros**. O treinamento também sugere quais entre as características descritas na seção 2.1.1 formam um conjunto ótimo de características para uma dada aplicação. O processo de determinação deste conjunto é chamado **seleção ótima de características**.

A seleção automática de parâmetros (seção 2.1.3.1) e a seleção ótima de características (seção 2.1.3.2) encontram os parâmetros ótimos do processo de classificação para uma borda de objeto em uma dada aplicação. Estes parâmetros são fixos e reutilizados para classificação

automática de outras fatias da cena durante a segmentação. Uma variação, porém, pode ser permitir que estes parâmetros sejam recalculados automaticamente tendo como base o resultado da segmentação de fatias anteriores. Este processo é chamado **treinamento adaptativo** (seção 2.1.3.3).

2.1.3.1 Seleção Automática de Parâmetros

Para treinar uma borda, o usuário utiliza o cursor do *mouse* para traçar manualmente segmentos típicos desta borda em **C**. O traçado manual pode ser realizado em uma ou mais fatias da cena, embora um ou dois segmentos em uma dada fatia sejam normalmente suficientes. Estes segmentos devem ser traçados consistentemente em um único sentido (horário ou anti-horário) definindo uma orientação para a borda desejada. O usuário deve evitar segmentos atípicos, tais como partes mal definidas da borda, e escolher apenas segmentos que representam a maior parte da borda desejada. Em seguida, para cada característica f_i , $1 \leq i \leq 14$, os seguintes parâmetros estatísticos são calculados: **moda** m_{f_i} , **desvio padrão em torno da moda** σ_{f_i} , **valor mínimo** \min_{f_i} e **valor máximo** \max_{f_i} .

Estes parâmetros estatísticos são associados aos parâmetros das transformadas $c_j(f_i)$, $j = 1, 2, 3$ e $i = 1, 2, \dots, 14$, descritas na seção 2.1.2 da seguinte forma: para $c_1(f_i)$, $l_1 = \min_{f_i}$, $h_1 = \max_{f_i}$ e $\mu_1 = m_{f_i}$, para $c_2(f_i)$, $l_2 = \min_{f_i}$, $h_2 = \max_{f_i}$, $\mu_2 = m_{f_i}$ e $\sigma_2 = \sigma_{f_i}$, e para $c_3(f_i)$, $l_3 = \min_{f_i}$, $h_3 = \max_{f_i}$ e $\mu_3 = m_{f_i}$. Selecionado qualquer par de característica f_i , $1 \leq i \leq 14$, e transformada c_j , $1 \leq j \leq 3$, valores baixos de custo $c_j(f_i(b))$ são associados aos bels b em **C** que têm valor de propriedade $f_i(b)$ próximo ao valor de pico da densidade de probabilidade da característica f_i medido para a borda desejada. Valores altos de custo são associados aos demais bels b de **C** que não satisfazem esta condição.

2.1.3.2 Seleção Ótima de Características

Visto que nem todas as características descritas na seção 2.1.1 são adequadas para uma dada aplicação, o objetivo desta seção é facilitar o trabalho do usuário de selecionar o subconjunto ótimo destas características para uma dada aplicação. O método proposto determina quais entre as características disponíveis têm maior probabilidade de classificação dos elementos da borda desejada em uma vizinhança dos segmentos de treinamento. Este método é descrito a seguir.

Seja B o conjunto dos bels da cena **C** indicado via treinamento, N o conjunto de todos os bels em uma pequena vizinhança dos bels em B , $\bar{B} = N - B$, e \min_{f_i} e \max_{f_i} os

valores de mínimo e máximo da característica f_i para os bels em B .

Seja $|X|$ a cardinalidade de qualquer conjunto X , seja $F = \{f_1, f_2, \dots, f_m\}$ (e.g. $m = 14$) um conjunto de características a serem selecionadas e $X = \{f_{i_1}, f_{i_2}, \dots, f_{i_j}\}$ qualquer subconjunto não-vazio de F . Considere H_X o hipercubóide definido pelos j intervalos $[min_{f_i}, max_{f_i}]$:

$$H_X = [min_{f_{i_1}}, max_{f_{i_1}}] \times \dots \times [min_{f_{i_j}}, max_{f_{i_j}}]. \quad (2.19)$$

O processo de seleção ótima de características tem por meta selecionar o subconjunto X de características que garante o menor número de bels em \bar{B} cujos valores de característica estão em H_X . Isto significa que a combinação ideal de características é aquela que resulta um conjunto de bels tão próximo quanto for possível de B (no melhor caso igual a B). Para facilitar esta seleção ótima, a seguinte função é maximizada para todos subconjuntos X de F :

$$P_B(X) = \frac{|B|}{|B| + |\bar{B}_{H_X}|}, \quad (2.20)$$

onde

$$\bar{B}_{H_X} = \{b \in \bar{B} \mid (f_{i_1}(b), f_{i_2}(b), \dots, f_{i_j}(b)) \in H_X\}. \quad (2.21)$$

Existem $2^m - 1$ possíveis subconjuntos não-vazios de F (visto que o conjunto F tem m elementos). Dados B , F e uma cena \mathbf{C} , $P_B(X)$, portanto, tem $2^m - 1$ elementos. Para pequenos valores de m , o conjunto X que maximiza $P_B(X)$ pode ser determinado por busca exaustiva. Neste trabalho, esta busca é realizada em poucos segundos logo após o usuário selecionar os segmentos típicos da borda para treinamento.

Determinado o conjunto ótimo de características, cada característica deste conjunto é convertida para valores de custo usando uma das transformadas descritas na seção 2.1.2 e estes valores de custo são combinados em um valor de custo conjunto usando a equação 2.18. Os parâmetros *default* para este processo são: transformada c_2 (complementar da Gaussiana) para todas características f_i , $i = 1, 2, \dots, m$, pesos $w_i = 1$ para todas as características independentes da orientação e pesos $w_i = 2$ para todas as características f_i sensíveis à orientação da borda.

2.1.3.3 Treinamento Adaptativo

Conforme descrito na seção anterior, o treinamento é realizado visando classificar uma borda de objeto em uma dada fatia. Isto não significa que o usuário tem que treinar esta

borda nas outras fatias da cena. As características, transformadas e parâmetros escolhidos na classificação de uma fatia são normalmente adequados para classificar automaticamente as outras fatias da cena. Em algumas situações, porém, as propriedades que definem a borda do objeto podem variar de uma fatia para outra requerendo um novo treinamento ou um simples ajuste de parâmetros da classificação. Como esta variação é normalmente gradativa e a borda é delineada fatia por fatia nos métodos *live*, o contorno que representa a borda na fatia anterior pode ser usado como traçado de treinamento para classificação da fatia atual, formando assim um processo de treinamento adaptativo.

O treinamento adaptativo é automático, mas tem algumas desvantagens. Ele aumenta o tempo de processamento e nem sempre garante melhorar a classificação. No caso de bordas com muitas partes mal definidas, o contorno gera muitas amostras de treinamento atípicas para descrever a borda desejada podendo dispersar a curva das transformadas. A opção de treinamento adaptativo é portanto facultativa nos métodos *live*.

2.1.4 Comentários

O processo de classificação pode ser resumido da seguinte forma. Dada uma cena 2D (3D ou 4D), o usuário inicialmente seleciona uma fatia qualquer da cena para treinamento da borda de interesse. A figura 2.4, por exemplo, ilustra o processo de treinamento da borda da traquéia em uma fatia obtida por CT. A figura 2.4a mostra esta fatia e a região de interesse escolhida para treinamento. Esta região é aumentada na tela do computador para que o usuário possa traçar com precisão os segmentos típicos da borda desejada. A figura 2.4b mostra o segmento desenhado pelo usuário. Em seguida, os processos de seleção automática de parâmetros e seleção ótima de características são executados e o custo conjunto é calculado usando a equação 2.18. O custo de qualquer característica e o custo conjunto são avaliados pelo usuário através de uma imagem de custo que é gerada associando a cada pixel um valor de brilho proporcional à média aritmética dos valores de custo dos *bels* que pertencem a este pixel. A figura 2.4c mostra a imagem de custo conjunto para as características f_1, f_2 e f_4 resultantes do processo de seleção ótima de características. Neste caso, apenas as características $f_i, i = 1, 2, \dots, 6$, participaram da seleção ótima. Note que valores baixos de custo (pixels escuros) são associados aos pixels da borda desejada. Valores altos de custo (pixels mais claros) são associados aos demais pixels da cena, exceto para aqueles que pertencem a outras bordas/regiões com propriedades similares à borda desejada. O usuário pode aceitar a classificação sugerida pelo treinamento ou selecionar outras características, selecionar outras transformadas, modificar os parâmetros das transformadas e modificar os

pesos da equação 2.18. Ajustados os parâmetros de classificação para uma fatia inicial, estes parâmetros são usados para classificar automaticamente qualquer outra fatia da cena antes de iniciar a segmentação.

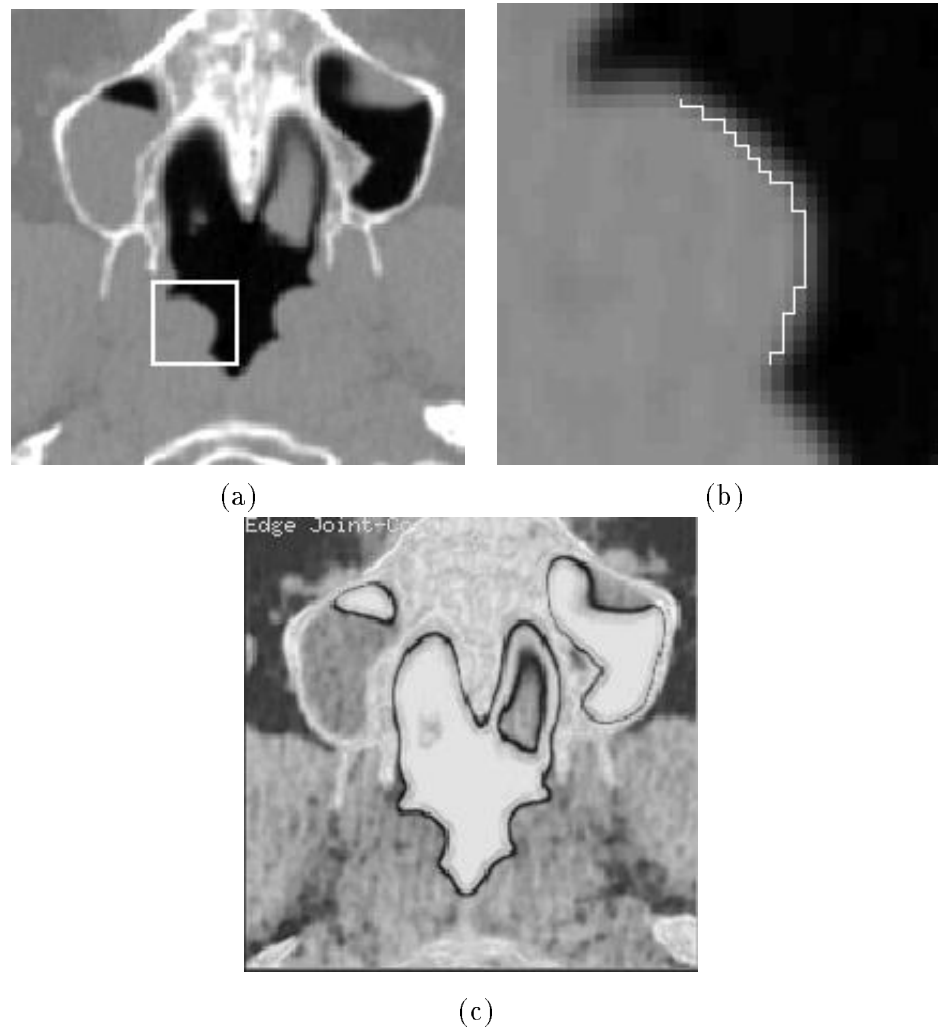


Figura 2.4: A figura ilustra o processo de treinamento da borda de uma secção transversal da traquéia. (a) Uma fatia transversal da traquéia de um paciente obtida por CT mostrando a região de interesse para treinamento. (b) Segmento da borda de interesse desenhado pelo usuário. (c) Imagem do custo conjunto gerada após a classificação usando as características f_1 , f_2 e f_4 .

As características usadas no exemplo da figura 2.4 são independentes da orientação da borda. Para entender qual a influência das características sensíveis à orientação no processo de classificação, o usuário pode visualizar as imagens de custo separando os bels com orientação oeste e sul dos bels com orientação leste e norte (ver figura 2.2). Neste caso duas imagens

de custo são geradas. A primeira imagem associa a cada pixel a combinação aritmética dos valores de custo dos bels oeste e sul deste pixel e a segunda imagem associa a cada pixel a combinação aritmética dos valores de custo dos bels leste e norte deste pixel. A figura 2.5 mostra estas imagens para uma fatia da junta de um pulso obtida por MRI. A fatia original que contém a borda do osso em segmentação é apresentada na figura 2.5a. A orientação da borda é definida via treinamento no sentido anti-horário e o custo conjunto é gerado usando as características f_{12} , f_{13} e f_{14} sensíveis a esta orientação. As figuras 2.5b e 2.5c mostram as duas imagens de custo conjunto resultantes do processo de classificação. Note que custos mais baixos são atribuídos apenas às partes superior (devido aos bels com orientação oeste) e lateral esquerda (devido aos bels com orientação sul) da borda na figura 2.5b. Na figura 2.5c os custos mais baixos são atribuídos às partes complementares (parte inferior devido aos bels com orientação leste e parte lateral direita devido aos bels com orientação norte).

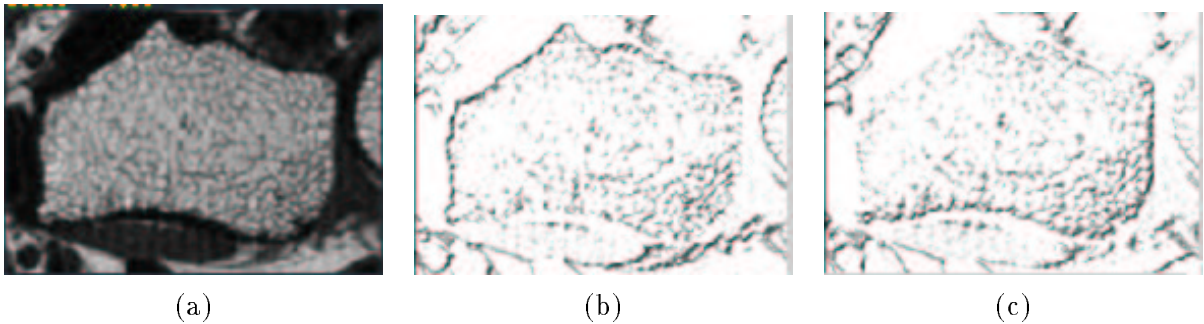


Figura 2.5: A figura mostra como o custo conjunto incorpora a informação sobre a orientação da borda. A orientação é assumida no sentido anti-horário e o custo conjunto é gerado usando as características f_{12} , f_{13} e f_{14} sensíveis à orientação da borda. (a) Fatia transversal da junta do pulso de um paciente obtida por MRI. (b) Imagem do custo conjunto gerada usando apenas os valores de custo de bels com orientação oeste e sul. Note que custos mais baixos são atribuídos às partes superior e lateral esquerda da borda. (c) Imagem do custo conjunto gerada usando apenas os valores de custo de bels com orientação leste e norte. Custos mais baixos são atribuídos às partes inferior e lateral direita da borda.

O processo de classificação recém descrito pode ser integrado a outras técnicas de segmentação de imagens. Técnicas baseadas em contornos ativos [9, 10, 18, 37, 41, 42, 44, 52, 88], por exemplo, podem usar a classificação para representar a energia externa do modelo. Este processo de classificação também permite que outras características de detecção de borda [49, 87, 91] sejam facilmente incorporadas ao método. Além do uso de características que descrevem a borda do objeto, uma pequena modificação no método de classificação pode permitir o uso de características para descrição de regiões da imagem, tais como medidas de

textura [30, 62, 85, 89]. Neste caso, um treinamento separado pode ser utilizado para marcar regiões de pixel no interior (ou exterior) do objeto de interesse. Como a idéia é detectar a borda do objeto, aos bels de pixels destas regiões devem ser associados altos valores de custo. Isto pode ser feito usando o complementar das transformadas apresentadas na seção 2.1.2. O resultado é uma função de custo que pode ser combinada na equação 2.18 com as funções de custo das outras características.

No caso de imagens muito ruidosas, o uso de filtros de processamento de imagens, antes do processo de segmentação, para reduzir ou eliminar ruído é uma boa estratégia para garantir uma função de custo de boa qualidade.

Pela teoria de Bayes [21, 28], o erro de um processo de classificação pode ser minimizado se as densidades de probabilidade dos valores de cada característica para cada classe (objeto e não objeto) e a probabilidade com a qual as classes ocorrem são conhecidas. Considerando esta observação, uma forma intuitiva de mapear uma dada característica em custo é usar como transformada de característica o complementar da função densidade de probabilidade da característica para a borda desejada. As transformadas de característica apresentadas na seção 2.1.2 podem ser vistas como o complementar de um modelo unimodal destas densidades de probabilidade, onde os parâmetros do modelo são estimados via treinamento supervisionado [21]. No caso da transformada c_2 , especificamente, a moda e o desvio padrão em torno da moda correspondem respectivamente às estimativas de máxima verossimilhança da média e do desvio padrão de uma densidade de probabilidade Gaussiana. As transformadas da seção 2.1.2 se diferenciam, portanto, no tipo de espalhamento que a distribuição de probabilidade tem em torno da moda. Em situações práticas, uma borda de objeto em \mathbf{C} pode ser caracterizada por múltiplas interfaces de diferentes contrastes com o exterior do objeto na cena e, portanto, a densidade de probabilidade de uma dada característica na borda pode ter múltiplas modas. Neste caso, transformadas de característica com múltiplos pontos de mínimo poderiam ser usadas e o treinamento seria realizado traçando segmentos de cada interface separadamente. No entanto, apesar da simplicidade do modelo unimodal, os resultados da classificação têm sido satisfatórios mesmo em situações complexas de segmentação como pode ser observado nos exemplos dados no decorrer deste trabalho.

2.2 Conectividade

Considere o grafo direcionado $G = (V, E)$ apresentado na figura 2.6a. $G = (V, E)$ consiste de um conjunto finito V de $|V|$ vértices (ou nós) e um conjunto E de $|E|$ arcos.

Cada vértice $v \in V$ pode ter até quatro vértices $u_k \in V$, $k = 1, 2, 3, 4$, distintos e vizinhos 4-adjacentes, formando o conjunto $A(v)$ dos vértices adjacentes ao vértice v em G (figura 2.6b). Um arco $e = (v, u)$ em E é um par ordenado de vértices adjacentes em G que inicia no vértice v , termina no vértice $u \in A(v)$ e tem associado um valor de comprimento $l(v, u)$, ou $l(e)$, no intervalo $[0, 1]$. Entre dois vértices adjacentes, $v, u \in V$, existem dois possíveis arcos, $(u, v), (v, u) \in E$, onde $l(u, v)$ pode ser diferente de $l(v, u)$. Um **caminho** $P = (e_1, e_2, \dots, e_n)$ em G é formado por uma sequência de arcos conexos $e_i = (v_{i-1}, v_i)$, $i = 1, 2, \dots, n$, onde o vértice final de um arco é o vértice inicial do próximo arco (figura 2.6a, onde $n = 5$). P é dito um **caminho simples** em G quando não repete nenhum vértice, $v_i \neq v_j$ para todo $i \neq j$. O **comprimento** de um caminho P é definido como a soma do comprimento de seus arcos $d(P) = l(e_1) + l(e_2) + \dots + l(e_n)$. $P = \langle v, v' \rangle$ é dito o **caminho mais curto** de um vértice $v \in V$ para um vértice $v' \in V$ se, considerando o comprimento de todos os possíveis caminhos simples de v a v' , $d(P)$ é o comprimento mínimo.

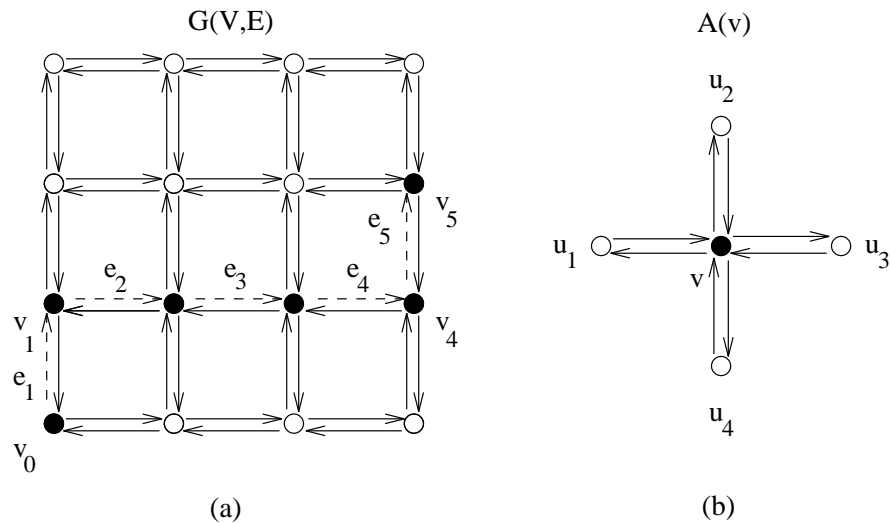


Figura 2.6: (a) Um grafo $G(V, E)$ definido para a cena \mathbf{C} após classificação. Um caminho $P = (e_1, e_2, \dots, e_5)$ em G entre os vértices v_0 e v_5 . (b) Cada vértice $v \in V$ define um conjunto $A(v) = \{u_1, u_2, u_3, u_4\}$ de no máximo 4 vértices vizinhos 4-adjacentes.

A classificação da cena $\mathbf{C} = (C, g)$ é vista como uma transformação de $\mathbf{C} = (C, g)$ em $G = (V, E)$, onde os vértices de G correspondem aos vértices dos pixels de \mathbf{C} e os arcos de G correspondem aos bels (arestas de pixel orientadas) de \mathbf{C} . O comprimento $l(e)$ de cada arco e de G corresponde ao custo conjunto $c(b)$ associado a cada bel b de \mathbf{C} na classificação. A idéia básica da conectividade consiste em transformar o problema de encontrar o **segmento ótimo** (ou segmento de custo mínimo) da borda do objeto em \mathbf{C} , que vai de um vértice v_s a

um vértice v_e marcados sobre a borda, no problema de encontrar o **caminho mais curto** no grafo G partindo de v_s e chegando a v_e .

O problema de encontrar o caminho mais curto (ou caminho ótimo) entre dois vértices de um grafo é referido na literatura de otimização e inteligência artificial [2, 17, 53] como *the shortest path problem*. Existe uma relação enorme de referências bibliográficas que propõem soluções para este problema. Uma pesquisa baseada em trezentas e uma referências classifica os algoritmos que encontram o caminho ótimo em um grafo em três categorias [19]. A primeira categoria está relacionada com o tipo de questão que envolve o grafo de entrada, tal como, qual é o caminho mais curto que parte de um vértice x para um vértice y passando por um vértice z . Dentro desta categoria existem dois tipos de algoritmo de interesse para os métodos *live*: (i) algoritmos que acham todos os caminhos ótimos de um **ponto de inicial para todos os outros pontos** do grafo e (ii) algoritmos que acham o caminho ótimo de um **ponto a outro** do grafo. A segunda categoria classifica os algoritmos pelas características salientes do grafo, como por exemplo, se o grafo é direcionado ou não direcionado, se o grafo é denso ou esparso e se os pesos $l(u, v)$ dos arcos de G são apenas valores positivos ou também assumem valores negativos. Nesta categoria o grafo $G = (V, E)$ é direcionado, pois os arcos $(v, u) \neq (u, v)$ para todo $(v, u), (u, v) \in E$, e possui apenas valores positivos de custo associados a cada arco. G é também considerado esparso¹, pois o número de arcos $|E|$ é normalmente bem menor que o quadrado do número de vértices $|V|^2$ [2, 17]. A terceira categoria classifica os algoritmos pela estratégia utilizada para obter o caminho ótimo. Nesta categoria, os algoritmos de interesse para os métodos *live* são classificados em dois grupos de algoritmos iterativos [2]: (i) *label-setting* e (ii) *label-correcting*. Algoritmos *label-setting* são aqueles que a cada iteração associam a um vértice do grafo o valor final do custo do caminho ótimo partindo do vértice inicial. Algoritmos *label-correcting* consideram todos os valores associados aos vértices do grafo temporários até a última iteração.

Nas seções 2.2.1 e 2.2.2 são apresentados os algoritmos LW_1 [26] e LW_2 [27] implementados neste trabalho. O algoritmo LW_1 é um *label-correcting* e o algoritmo LW_2 é um *label-setting*. Ambos garantem solução ótima global para resolver os problemas de encontrar (i) todos os caminhos ótimos partindo de um ponto e (ii) o caminho ótimo de um ponto a outro no grafo. Estes algoritmos utilizam a estratégia de programação dinâmica [4] e são descritos a seguir.

¹Um grafo é considerado denso quando $|E| = \omega(|V|^2)$ onde ω é uma função quadrática de $|V|$. No caso do grafo G tem-se para uma cena com $(n - 1) \times (m - 1)$ pixels, $|V|^2 = (nm)^2$ e $|E| = 4nm - 2n - 2m$. A função ω é uma função linear de $|V|$.

2.2.1 Algoritmo LW_1

Nos métodos *live*, quando o usuário está guiando o processo de delinear a borda do objeto, um ponto inicial v_s (vértice de pixel) é dado sobre a borda do objeto e o ponto final v_e pode ser qualquer outro ponto da cena em que o usuário posicione o cursor do *mouse*. Nestas situações, o efeito *live* que dá origem ao nome dos métodos ocorre porque todos os caminhos ótimos possíveis que vão do ponto inicial a qualquer outro ponto da cena são pré-calculados e, portanto, eles são apresentados na tela do computador instantaneamente para qualquer posição do cursor sobre a cena (ver capítulo 3).

Dado uma cena \mathbf{C} , um grafo $G_{in}(V_{in}, E_{in})$ é gerado pela classificação desta cena como descrito anteriormente. Selecionado um vértice inicial v_s de G_{in} , o algoritmo LW_1 encontra todos os caminhos ótimos que vão de v_s a qualquer outro vértice de G_{in} e grava estes caminhos em um grafo de saída $G_{out}(V_{out}, E_{out})$. Para qualquer outro vértice v_e de G_{in} , o caminho ótimo entre v_s e v_e é traçado apenas percorrendo os arcos de G_{out} conforme descrito abaixo.

Algoritmo LW_1

Entrada: O grafo $G_{in} = (V_{in}, E_{in})$ onde V_{in} é o conjunto dos vértices dos pixels da cena \mathbf{C} , E_{in} é o conjunto dos bels b de \mathbf{C} e a cada bel b está associado um valor de custo conjunto $c(b)$ no intervalo $[0, 1]$; um vértice inicial $v_s \in V_{in}$.

Saída: O grafo $G_{out} = (V_{out}, E_{out})$ onde V_{out} é o conjunto dos vértices dos pixels da cena \mathbf{C} e E_{out} é o conjunto das arestas de pixel de \mathbf{C} . Neste grafo, entre cada dois vértices adjacentes de G_{out} existe nenhum ou no máximo um arco orientado que faz parte de todos os caminhos ótimos que contém esta aresta de pixel.

Estruturas de Dados Auxiliares: Uma matriz 2D cujos elementos são os vértices dos pixels da cena \mathbf{C} e a cada vértice v é associado o “custo acumulado” $cc(v)$ representando o custo total temporário do caminho ótimo que parte de v_s para o vértice v ; uma fila Q de vértices para os quais a informação do caminho ótimo partindo de v_s está para ser atualizada.

início

1. inicialize $cc(v) = \infty$ para todos os vértices v de \mathbf{C} , $cc(v_s) = 0$ e G_{out} como um grafo não direcionado;
 2. coloque os vértices de $A(v_s)$ (vizinhos 4-adjacentes de v_s) na fila Q ;
 3. *enquanto* Q não estiver vazia *faça*
 - a. remova um vértice v de Q ;
 - b. encontre o conjunto dos vértices v' de $A(v)$ tal que $cc(v') < cc(v)$;
 - c. *se* este conjunto não for vazio *então*
 - (i) *para* cada vértice v' deste conjunto, calcule $cc_{\min} = \min_{v'}[cc(v') + c(b')]$ onde b' é o bel que vai de v para v' e $c(b')$ é o custo conjunto de b' , e encontre o vértice v_j (e bel b_j) entre os v' para o qual o mínimo ocorreu;
 - (ii) *se* $cc_{\min} < cc(v)$ *então*
 - a. faça $cc(v) = cc_{\min}$ e associe a direção do bel b_j , que vai do vértice v para v_j , à aresta de pixel correspondente em G_{out} .
 - b. coloque os vértices v_k de $A(v)$ tal que $cc(v_k) > cc(v)$ e $v_k \notin Q$ em Q ;
- fim de se;*
- fim de se;*
- fim de enquanto;*
4. dado qualquer vértice v_e de \mathbf{C} , trace recursivamente o caminho ótimo seguindo a direção das arestas de pixel de G_{out} que parte de v_e até encontrar v_s , e apresente este caminho na tela do computador;

fim

Nas etapas 2 e 3c(ii)b, vértices de $A(v)$ são colocados na fila Q sempre no sentido horário iniciando no vértice u_1 para o vértice u_4 (ver figura 2.6b). Na etapa 3a, sempre o primeiro vértice de Q é removido da fila. Este tipo de estratégia é conhecida como FIFO (*first in first out*). Note que para qualquer vértice v_e de \mathbf{C} , o caminho ótimo partindo de v_s para v_e é na verdade traçado no sentido contrário que vai de v_e para v_s . Por isso, este tipo de algoritmo também é referido como algoritmo que acha os caminhos ótimos de todos os vértices do grafo a um vértice destino [19]. O algoritmo LW_1 usa a estratégia de busca em largura (*breadth-first*) que sempre garante o caminho ótimo global [17, 53].

A figura 2.7 ilustra um exemplo do comportamento deste algoritmo. A figura 2.7a mostra o grafo de entrada G_{in} onde valores inteiros de custo conjunto de 0 a 10 são associados aos arcos, os vértices são representados por círculos claros e o vértice inicial v_s é representado pelo círculo escuro. As figuras de 2.7b a 2.7f mostram a matriz 2D de custos acumulados e o grafo de saída G_{out} combinados em uma só representação para diferentes iterações do algoritmo. Nestas figuras, o próximo vértice a sair da fila é sempre representado pelo círculo parcialmente escuro. A figura 2.7b mostra a situação inicial antes da primeira iteração. As figuras 2.7c e 2.7d mostram respectivamente a situação após a primeira e a segunda iteração. A figura 2.7e mostra a situação após sete iterações e a figura 2.7f mostra o resultado final após dezoito iterações.

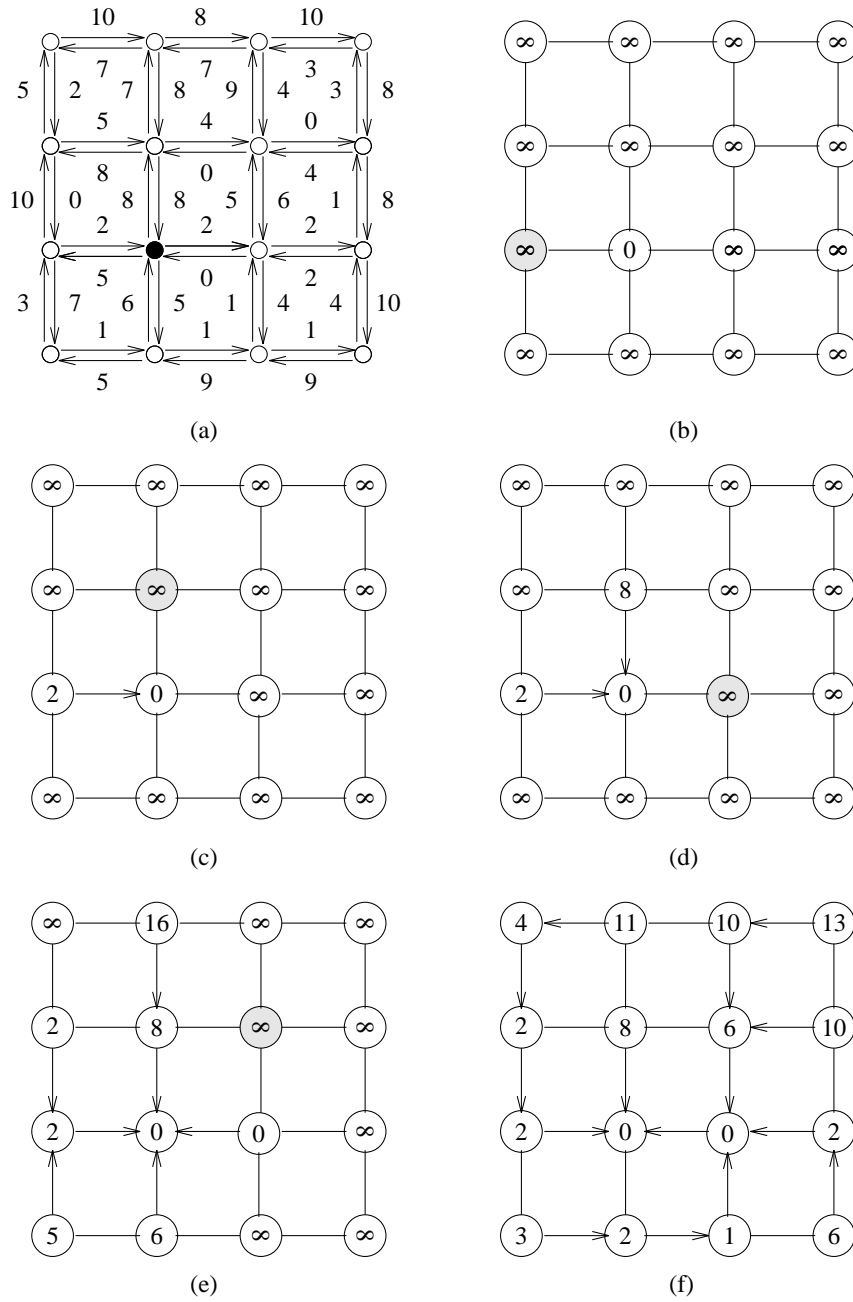


Figura 2.7: A figura ilustra o algoritmo LW_1 . (a) Um grafo de entrada representando uma cena C após classificação, onde valores de custo conjunto de $[0,10]$ são associados aos bels de C e o vértice inicial é representado pelo círculo escuro. As figuras de (b) a (f) ilustram respectivamente a situação do grafo de saída antes da primeira iteração, após a primeira iteração, após a segunda iteração, após a sétima iteração e após dezoito iterações. Nestas figuras, o próximo vértice a sair da fila é sempre representado pelo círculo parcialmente escuro.

Nos métodos *live*, idéias que minimizam o tempo total do usuário sem comprometer o sucesso da segmentação são sempre bem vindas. O uso de um valor de limiar T aplicado ao custo acumulado $cc(v)$, por exemplo, é uma boa idéia para reduzir o tempo de processamento do algoritmo LW_1 . Esta modificação pode ser implementada logo após a etapa 3a da seguinte forma: se $cc(v) > T$ e $cc(v) \neq \infty$, então as etapas 3b e 3c não precisam ser executadas. Para uma dada aplicação, T deve ser escolhido suficientemente maior do que o pior custo total possível para o contorno que descreve a borda do objeto. É óbvio que esta seleção heurística pode não garantir um caminho ótimo para todos os pontos do grafo, mas deve garantir em uma região de interesse representada por uma faixa de pontos em torno da borda do objeto. A utilidade prática desta idéia e de outras que reduzem o tempo de processamento de LW_1 é discutida no capítulo 3 para situações reais.

2.2.2 Algoritmo LW_2

O algoritmo LW_2 é baseado em algumas modificação do algoritmo de Dijkstra [20]. Este algoritmo sempre termina seu processamento em $n - 1$ iterações, onde n é o número de vértices de pixel da cena \mathbf{C} . O algoritmo como apresentado abaixo encontra todos os caminhos ótimos que partem de um ponto inicial v_s .

Algoritmo LW_2

Entrada: O grafo $G_{in} = (V_{in}, E_{in})$ onde V_{in} é o conjunto dos vértices dos pixels da cena \mathbf{C} , E_{in} é o conjunto dos bels b de \mathbf{C} e a cada bel b está associado um valor de custo conjunto $c(b)$ no intervalo $[0, 1]$; um vértice inicial $v_s \in V_{in}$.

Saída: O grafo $G_{out} = (V_{out}, E_{out})$ onde V_{out} é o conjunto dos vértices dos pixels da cena \mathbf{C} e E_{out} é o conjunto das arestas de pixel de \mathbf{C} . Entre cada dois vértices adjacentes em G_{out} existe nenhum ou no máximo um arco orientado que faz parte de todos os caminhos ótimos que contém esta aresta de pixel.

Estruturas de Dados Auxiliares: Uma matriz 2D cujos elementos são os vértices dos pixels da cena \mathbf{C} e a cada vértice v é associado "o custo acumulado" $cc(v)$ representando o custo total do caminho ótimo que parte de v_s para o vértice v ; uma fila Q de vértices ordenados por custo acumulado; uma lista L dos vértices já processados.

início

1. inicialize $cc(v) = \infty$ para todos os vértices v de \mathbf{C} , $cc(v_s) = 0$ e G_{out} como um grafo não direcionado;
2. coloque o vértice v_s na fila Q ;
3. *enquanto* Q não estiver vazia *faça*
 - a. remova um vértice v de Q e coloque v em L ;
 - b. *para* cada vértice v' de $A(v)$ (vizinhos 4-adjacentes de v) tal que $v' \notin L$;
 - (i) calcule $cc_{tmp} = cc(v) + c(b')$ onde b' é o bel que vai de v' para v e $c(b')$ é o custo conjunto de b' ;
 - (ii) *se* $cc_{tmp} < cc(v')$ *então*
 - a. faça $cc(v') = cc_{tmp}$ e associe a direção do bel b' , que vai do vértice v' para v , à aresta de pixel correspondente em G_{out} .
 - b. coloque v' em Q tal que os vértices de Q mantenham a ordem crescente de custo acumulado;

fim de se;

fim de para;

fim de enquanto;
4. dado qualquer vértice v_e de \mathbf{C} , trace recursivamente o caminho ótimo seguindo a direção das arestas de pixel de G_{out} que parte de v_e até encontrar v_s , e apresente este caminho na tela do computador;

fim

Na etapa 3a, o vértice v removido é sempre o vértice de custo acumulado $cc(v)$ mínimo em Q , pois a fila é mantida ordenada na ordem crescente de custo acumulado. Esta característica faz com que em situações onde o vértice final v_e é fixo, o algoritmo LW_2 pode terminar em um número bem menor de iterações se acrescentar a seguinte etapa entre 3a e 3b: após remover o vértice v de menor custo acumulado da fila Q (etapa 3a), se este vértice for o vértice v_e então executa-se a etapa 4. Esta modificação reduz de 7 a 20 vezes o tempo de processamento do algoritmo LW_2 , nos exemplos usados neste trabalho.

A figura 2.8 ilustra um exemplo comparativo do comportamento do algoritmo LW_2 com relação ao algoritmo LW_1 usando o mesmo grafo de entrada e o mesmo vértice inicial

v_s da figura 2.7a (ver figura 2.8a). As figuras de 2.8b a 2.8f mostram a matriz 2D de custos acumulados e o grafo de saída G_{out} combinados em uma só representação para as mesmas situações do exemplo 2.7. Nestas figuras, o próximo vértice a sair da fila é sempre representado pelo círculo parcialmente escuro. A figura 2.8b mostra a situação inicial antes da primeira iteração. As figuras 2.8c e 2.8d mostram respectivamente a situação após a primeira e a segunda iteração. A figura 2.8e mostra a situação após sete iterações e a figura 2.8f mostra o resultado final após quinze iterações. Note que se o vértice v_e for previamente determinado como sendo, por exemplo, o vértice do canto superior esquerdo do grafo G_{in} (figura 2.8a), o caminho ótimo de v_s a v_e é encontrado logo após a sétima iteração (figura 2.8e) e, com a modificação discutida no parágrafo anterior, o algoritmo não precisa continuar até a última iteração.

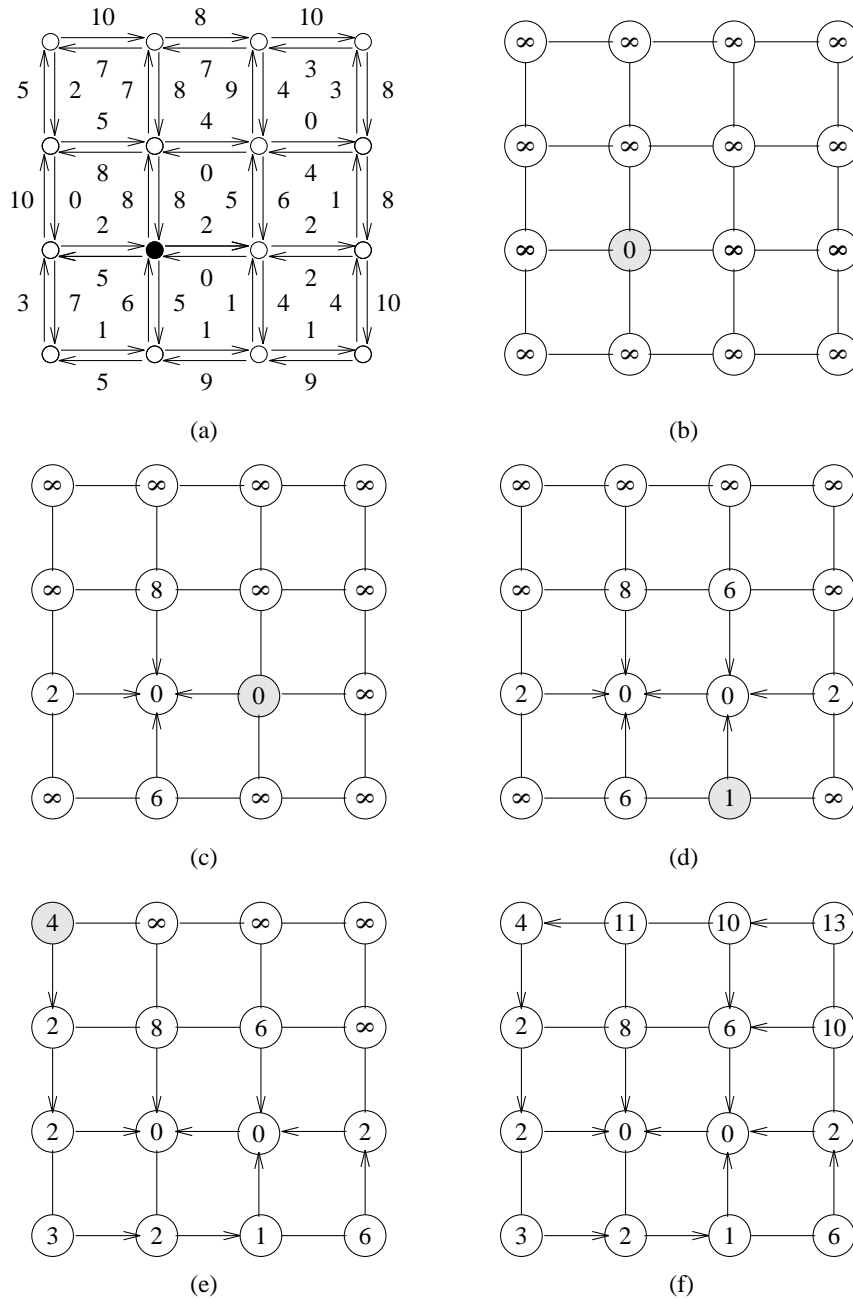


Figura 2.8: A figura ilustra o algoritmo LW_2 . (a) Um grafo de entrada representando uma cena C após classificação, onde valores de custo conjunto de $[0,10]$ são associados aos bels de C e o vértice inicial é representado pelo círculo escuro. As figuras de (b) a (f) ilustram respectivamente a situação do grafo de saída antes da primeira iteração, após a primeira iteração, após a segunda iteração, após a sétima iteração e após quinze iterações. Nestas figuras, o próximo vértice a sair da fila é sempre representado pelo círculo parcialmente escuro.

Observe que diferente da busca em largura ilustrada no exemplo da figura 2.7, o algoritmo LW_2 expande o grafo sempre na direção de custo acumulado mínimo caracterizando-o como um algoritmo guloso (*greedy algorithm*). Esta estratégia aplicada às funções de custo conjunto, tais como a da figura 2.4c, faz com que o processamento ocorra primeiro para vértices em uma certa faixa em torno da borda desejada e, portanto, para vértices v_s e v_e sobre esta borda, o algoritmo encontra o caminho ótimo sem precisar processar a maioria dos vértices da cena. Algoritmos gulosos nem sempre garantem o caminho ótimo, mas no caso do algoritmo LW_2 pode-se provar facilmente que o caminho ótimo é sempre garantido [17].

A idéia do valor de limiar T aplicado ao custo acumulado $cc(v)$ também pode ser usada no algoritmo LW_2 para agilizar o processamento. Neste caso, a etapa 3b(ii) deve ser modificada para testar se $cc_{tmp} < cc(v')$ e $cc(tmp) < T$, onde T é escolhido suficientemente maior do que o pior custo total possível para o contorno que descreve a borda do objeto.

2.2.3 Comentários

O algoritmo LW_1 apresenta duas desvantagens quando comparado ao LW_2 . A primeira é que para um número n de vértices de pixel da cena \mathbf{C} , este algoritmo pode requerer bem mais que $n - 1$ iterações para terminar. Isto se deve ao fato que um mesmo vértice pode entrar na fila Q mais de uma vez até assumir seu valor mínimo global de custo acumulado. A segunda característica é que mesmo fixando o vértice final v_e , o algoritmo não pode terminar seu processamento até que se esvazie a fila Q . Por outro lado, o algoritmo LW_2 requer que a fila Q esteja sempre ordenada acrescentando um aumento no tempo de processamento.

O tempo de processamento do algoritmo LW_2 depende principalmente da estrutura de dados utilizada para a fila Q e do mecanismo de ordenação dos vértices de Q . Em [2] são propostas várias formas de implementar o algoritmo de Dijkstra usando diferentes estruturas de dados. Apesar da análise de complexidade mostrar que para algumas estruturas a velocidade de processamento é bem maior do que para outras, as estruturas de dados devem ser apropriadas às características do grafo, tais como, a magnitude do valor associado ao custo dos arcos, o número de arcos, o número de vértices e a densidade do grafo. A implementação original de Dijkstra, por exemplo, continua sendo considerada a que tem tempo ótimo de execução para grafos muito densos [2]. Cenas 2D típicas em imagens médicas podem ter até 512×512 pixels resultando um grafo G com 262.144 vértices e 1.046.528 arcos. Este tipo de grafo é normalmente bem maior do que os grafos considerados na literatura de caminhos ótimos [19]. Encontrar a forma mais rápida de processar este tipo de grafo requer um esforço adicional em pesquisa. Neste trabalho, a implementação adotada para ambos algoritmos usa

uma estrutura de dados de fila cíclica para Q [17] e executa o processamento completo do grafo G_{out} em poucos segundos. Este tempo tem sido suficiente para que a resposta às ações do usuário nos métodos *live* seja obtida em tempo real.

A modificação do algoritmo de Dijkstra que dá origem ao algoritmo LW_2 só foi desenvolvida no final do trabalho e, portanto, a avaliação dos métodos *live* (capítulos 3 e 4) foi feita usando o algoritmo LW_1 . Experimentos comparando os algoritmos em diferentes imagens mostram que o algoritmo LW_2 é normalmente 2-3 vezes mais rápido do que o algoritmo LW_1 e pode ser bem mais rápido se o ponto v_e for fixo. Portanto, o algoritmo LW_2 é o algoritmo atualmente utilizado nos métodos *live* para ambas situações: (i) caminho ótimo entre dois pontos e (ii) todos os caminhos ótimos partindo de um ponto.

2.3 Conclusão

Neste capítulo foram descritos os processos de classificação e conectividade utilizados pelos métodos *live*. A classificação requer em média um minuto do tempo total do usuário para obter as características, transformadas e parâmetros que serão utilizados para classificar automaticamente todas as outras fatias da cena. Terminada a classificação, o problema de detecção de um segmento de borda ótimo em uma cena \mathbf{C} é formulado como um problema de busca do caminho ótimo (custo mínimo) entre dois pontos de um grafo direcionado. A solução deste problema descreve o mecanismo de conectividade entre os pontos da borda do objeto em \mathbf{C} .

Para concluir este capítulo, o exemplo da figura 2.9 ilustra a importância do treinamento no processo de classificação e conectividade para extrair segmentos ótimos da borda de objetos em imagens digitais. A figura 2.9a mostra a fatia original de um joelho obtida por CT. Antes do treinamento, os parâmetros *default* da classificação para este exemplo são: característica f_6 , transformada c_2 , com $h_2 = \mu_2 = \max_{f_6}$, $l_2 = 0$ e $\sigma_2 = \max_{f_6}/10$, onde \max_{f_6} é o valor máximo de f_6 em \mathbf{C} . Evidentemente, para altas magnitudes de gradiente são associados baixos valores de custo e para baixas magnitudes de gradientes são associados altos valores de custo. A figura 2.9b mostra a imagem de custo conjunto c resultante da classificação usando os parâmetros acima. Se o objetivo é segmentar a borda de menor contraste que está no meio da parte inferior da imagem, então para quaisquer dois pontos v_s e v_e selecionados sobre esta borda, espera-se que o caminho ótimo de v_s a v_e descreva a borda desejada. No entanto, como mostrado na figura 2.9c, este caminho toma uma rota alternativa sendo fortemente atraído pelas bordas de alto contraste representadas por ossos

e pele. Após o usuário traçar típicos segmentos da borda de baixo contraste, como indicado na figura 2.9d, o custo conjunto resultante do treinamento é mostrado na figura 2.9e. Note na figura 2.9e como todas as bordas de baixo contraste (indicadas pelas setas) obtêm custos significativamente mais baixos comparados com os custos da figura 2.9b. O caminho ótimo de v_s a v_e é agora fortemente atraído pela borda de baixo contraste desejada (figura 2.9f) e fortemente repelido pelas bordas de alto contraste como ilustrado na figura 2.9g.

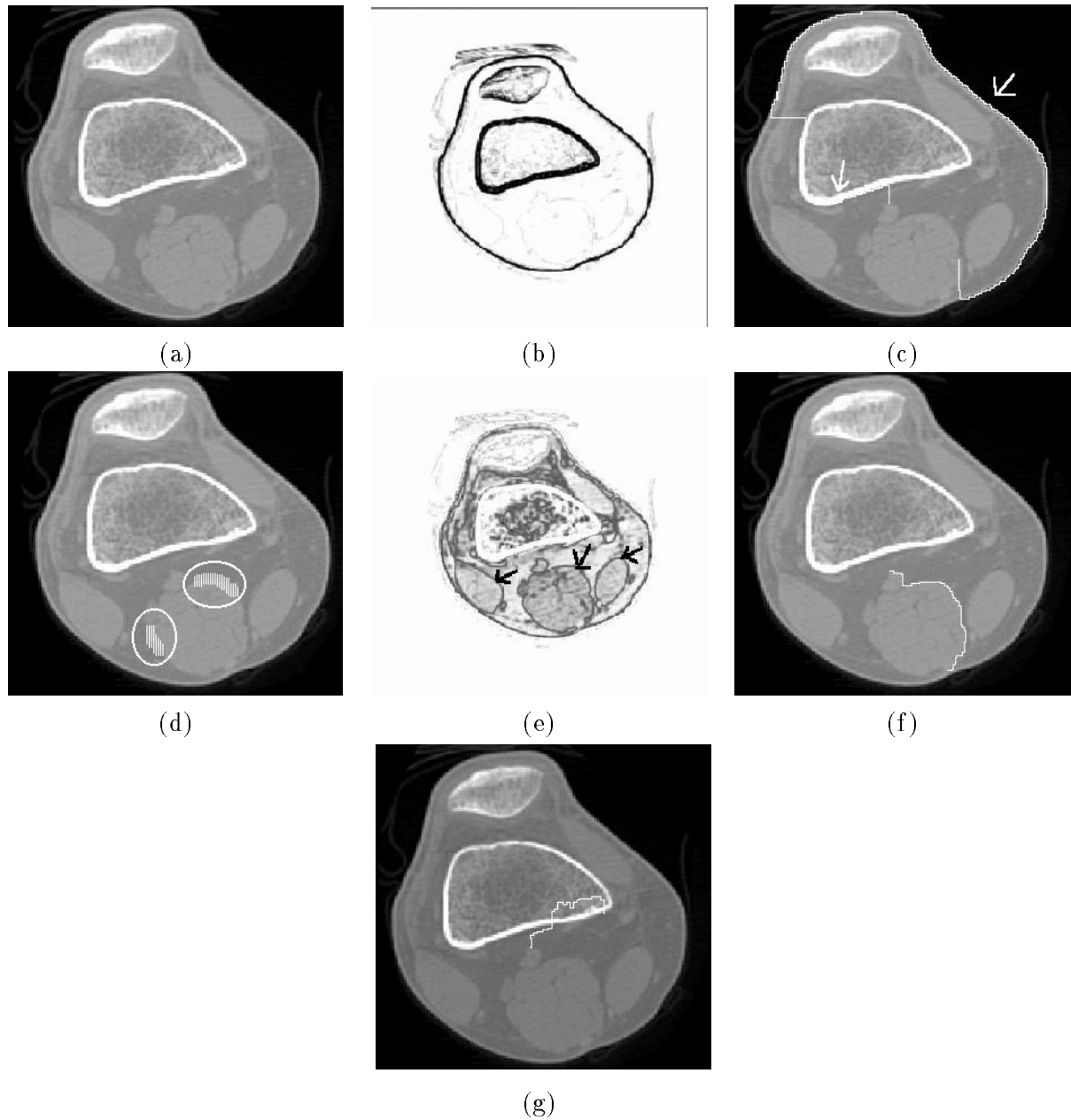


Figura 2.9: (a) Uma fatia do joelho obtida por CT. (b) Uma imagem mostrando o custo conjunto de todos os bels da cena (a), usando a característica f_6 , c_2 , $\mu_2 = h_2 = (\text{valor máximo de } f_6)$, $\sigma_2 = (\text{valor máximo de } f_6)/10$ e $l_2 = 0$. (c) Caminho ótimo sendo atraído pelas bordas de alto contraste. (d) Treinamento para as bordas de baixo contraste. (e) Função de custo conjunto após treinamento. (f) Caminho ótimo sendo atraído pelas bordas de baixo contraste. (g) Caminho ótimo sendo repellido pelas bordas de alto contraste.

O processo de classificação, como descrito neste capítulo, intende a definição de uma única borda da cena por vez. Entretanto, conforme pode ser observado neste exemplo, os parâmetros da classificação de uma dada borda podem servir para a classificação de outras bordas da cena (e.g. ossos e pele na figura 2.9b e bordas de baixo contraste na figura 2.9e).

Agora considere $V = \{v_0, v_1, \dots, v_n\}$ um conjunto de $n + 1$ vértices sobre a borda do objeto em \mathbf{C} , onde os vértices estão ordenados segundo a orientação da borda. A borda completa do objeto em \mathbf{C} é representada em G pela sequência $\langle v_i, v_{i+1} \rangle, i = 0, 1, \dots, n - 1$, de caminhos ótimos somada ao caminho ótimo $\langle v_n, v_0 \rangle$ para formar um contorno fechado, conectado e orientado em \mathbf{C} . A borda do objeto em \mathbf{C} é portanto ótima por partes. Os métodos *live-wire*, *live-lane* e *3D-live-wire* extraem bordas de objeto em \mathbf{C} desta forma. Eles se diferenciam em dois aspectos (i) no processo de seleção dos vértices v_i sobre a borda e (ii) no mecanismo de interação com o usuário que vai assistir à seleção destes vértices. A seguir, os métodos *live-wire* e *live-lane* são apresentados e avaliados no capítulo 3 e o método *3D-live-wire* é apresentado e avaliado no capítulo 4.

Capítulo 3

Live-Wire e Live-Lane

Existem várias maneiras de obter contornos fechados, conectados e orientados (compostos de arestas orientadas de pixel) em uma dada cena \mathbf{C} . Cada contorno é uma borda em potencial. Suponha que seja associado um custo para cada possível contorno fechado, conectado e orientado em \mathbf{C} , onde este custo é simplesmente a soma dos custos dos bels b deste contorno. O problema de encontrar o contorno com o menor custo total é um enorme problema de otimização, mas pode ser bastante simplificado se o usuário indicar um bel b_0 de \mathbf{C} que é parte do contorno procurado. O problema passa a ser o de encontrar o contorno fechado, conectado e orientado de custo mínimo em \mathbf{C} que contém b_0 . Para este problema, sejam v_s e v_e os vértices dos pixels que formam o bel b_0 em \mathbf{C} . Estes vértices indicam dois pontos da borda do objeto e o caminho de custo mínimo de v_s a v_e , calculado sem incluir o bel b_0 , junto ao bel b_0 formam uma borda ótima.

Infelizmente, a borda ótima detectada com esta forma simples de interação do usuário normalmente não coincide com a borda do objeto desejada. Exceto para uma situação ideal em que a cena tem apenas dois valores de cinza. A primeira razão para esta falha é a dificuldade de encontrar características de imagem e suas respectivas transformadas para gerar valores de custo conjunto que façam com que a borda ótima coincida com a borda desejada. A segunda razão é que o critério de otimalidade é baseado no custo total e, portanto, os algoritmos de conectividade favorecem caminhos de menor comprimento no grafo. Consequentemente, os algoritmos encontram pequenas voltas na vizinhança de b_0 . A figura 3.1 ilustra esta observação para a borda do osso tálus em uma fatia do pé obtida por MRI. Consequentemente, o usuário necessita selecionar mais vértices sobre a borda do objeto para haver sucesso na sua detecção.



Figura 3.1: A figura mostra uma fatia do pé de um paciente obtida por MRI, onde o objeto em evidência é o osso tálus. O caminho ótimo falha quando tenta encontrar a borda através de um único segmento de custo mínimo calculado usando v_s e v_e como os vértices de um bel inicial b_0 . O critério de optimalidade leva o algoritmo a encontrar pequenas voltas na vizinhança de b_0 .

No capítulo 2 foi mostrado que dado um conjunto $V = \{v_0, v_1, \dots, v_n\}$ de $n + 1$ vértices ordenados sobre a borda orientada do objeto em \mathbf{C} , o contorno fechado, conectado e orientado que representa a borda ótima pode ser encontrado resolvendo um problema de busca de caminhos ótimos $\langle v_i, v_{i+1} \rangle$ em um grafo direcionado G . Existem diferentes mecanismos possíveis de interação com o usuário para a seleção dos vértices de V . O número de vértices necessários varia com a aplicação. Bordas mal definidas na cena exigem maior número de vértices e, conseqüentemente, maior envolvimento e controle do usuário sobre o processo de seleção de vértices para garantir o sucesso da segmentação. Neste capítulo são apresentados dois métodos de segmentação de imagens, *live-wire* (seção 3.1) e *live-lane* (seção 3.2). Eles operam fatia por fatia e oferecem diferentes níveis de envolvimento e controle do usuário para seleção destes vértices. Estes métodos são avaliados e comparados com a segmentação manual na seção 3.3. A seção 3.4 conclui este capítulo discutindo os principais aspectos do comportamento dos métodos *live-wire* e *live-lane*.

3.1 *Live-Wire*

No método *live-wire*, o usuário seleciona inicialmente um vértice v_0 na borda do objeto desejado na cena \mathbf{C} usando o cursor do *mouse*. Para qualquer localização subsequente do cursor na cena, um caminho ótimo de v_0 para a posição atual v_1 do cursor é calculado e apresentado na tela do computador em tempo real. O usuário pode verificar se o caminho descreve a borda desejada movendo o cursor e portanto mudando v_1 (ver figura 3.2). Para minimizar o envolvimento do usuário no processo de segmentação, o segmento da borda desejado deve ser o maior possível obtido através deste caminho. Se o caminho ótimo é aceitável, o usuário deposita o vértice v_1 que subsequentemente passa a ser o vértice inicial e um novo segmento ótimo, que **não inclui** os segmentos previamente detectados, deve ser encontrado movendo o cursor para escolher o próximo vértice v_2 . O processo continua desta forma até que o usuário indique que deseja fechar o contorno, finalizando o processo de seleção de $n + 1$ vértices de V . Neste instante, o último segmento da borda é calculado entre o atual vértice v_n e o vértice inicial v_0 completando a borda do objeto na cena.

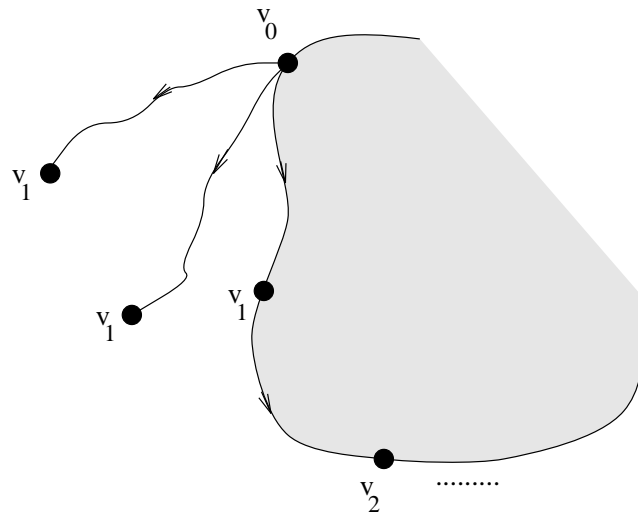


Figura 3.2: A figura ilustra o processo de segmentação usando *live-wire*.

É importante enfatizar que os vértices pelos quais passam os segmentos de *live-wire* previamente encontrados não são incluídos na determinação do segmento de *live-wire* atual. Isto evita que a curva que está sendo traçada cruze a si própria afetando a definição topológica de borda de objeto como uma curva de Jordan. É claro que situações podem ser criadas traçando os segmentos de *live-wire* de forma contorcida, como uma espiral por exemplo, fazendo com que não exista nenhum caminho possível entre os atuais v_i e v_{i+1} . Nestas

situações o usuário tem que escolher um vértice da curva traçada a partir do qual todo o resto da curva deve ser apagado e este vértice passa a ser o novo vértice inicial v_i . A mesma atitude é tomada quando o usuário não está satisfeito com a curva que está sendo traçada. Portanto, o usuário tem completo controle durante o processo para garantir a borda do objeto desejada.

Os métodos *live* são processos dinâmicos que infelizmente não podem ser adequadamente ilustrados através de figuras. Nas seções seguintes procura-se dar uma idéia do comportamento dinâmico do *live-wire* através de exemplos, análise de suas propriedades e possíveis variações.

3.1.1 Exemplos e Propriedades

A figura 3.3 mostra a mesma fatia da traquéia de um paciente obtida por CT e apresentada na figura 2.4. Neste exemplo, apenas a característica f_{14} com a transformada c_2 é utilizada para classificação. As figuras 3.3a e 3.3b indicam respectivamente o primeiro segmento, $\langle v_0, v_1 \rangle$, e o segundo segmento, $\langle v_1, v_0 \rangle$, necessários para completar a borda da traquéia nesta cena. O segmento $\langle v_0, v_1 \rangle$ é o maior segmento possível que descreve a borda desejada partindo de v_0 . A figura 3.3c ilustra esta observação mostrando um segmento $\langle v_s, v_e \rangle$ onde $v_s = v_0$ e v_e está um pouco além de v_1 na borda desejada. Os algoritmos de conectividade apresentados na seção 2.2 têm a tendência de optar por caminhos de menor comprimento no grafo G e, por isso, $\langle v_s, v_e \rangle$ não representa um segmento da borda.

A tendência dos algoritmos de conectividade de escolherem caminhos mais curtos também pode ser vista como uma vantagem em situações onde existe deficiência de informação da borda. A figura 3.4 ilustra esta observação para fatias da junta do pulso obtidas por MRI. Na figura 3.4a o objeto de interesse é o osso e na figura 3.4b o objeto de interesse é uma veia do pulso. As figuras 3.4a e 3.4b ilustram uma situação com muito ruído e outra com falha na borda respectivamente. Em ambas situações o segmento de *live-wire* passa imune pela região da borda onde há deficiência de informação.

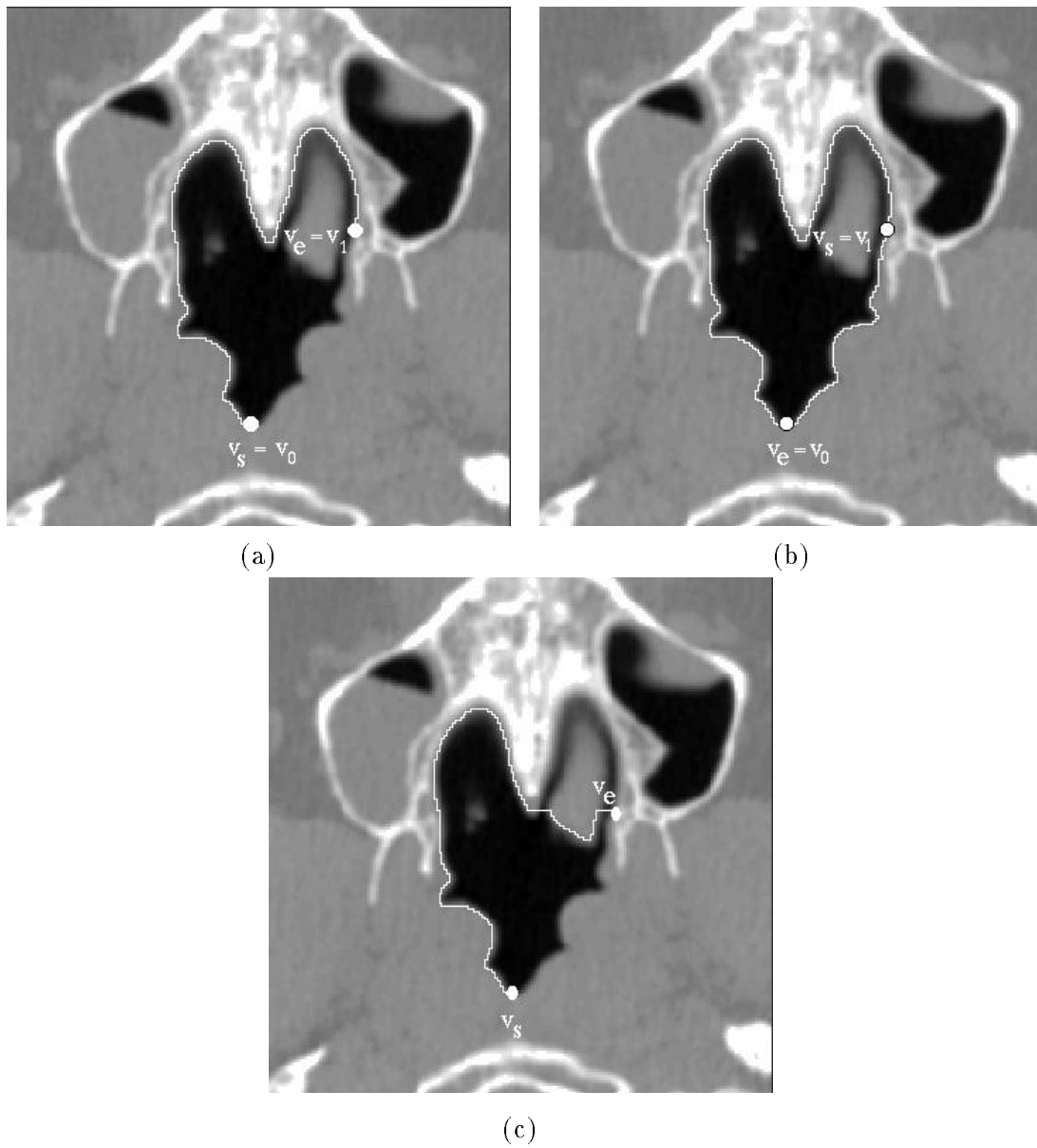


Figura 3.3: A figura ilustra a segmentação via *live-wire* de uma fatia transversal da traquéia obtida por CT. (a) O primeiro segmento $P_1 = \langle v_0, v_1 \rangle$ de *live-wire*. (b) O segundo segmento $P_2 = \langle v_1, v_0 \rangle$ de *live-wire*. (c) O segmento $\langle v_s, v_e \rangle$ ilustra a tendência do *live-wire* optar por caminhos mais curtos mostrando que P_1 é o segmento mais longo possível que descreve a borda desejada partindo de v_0 .

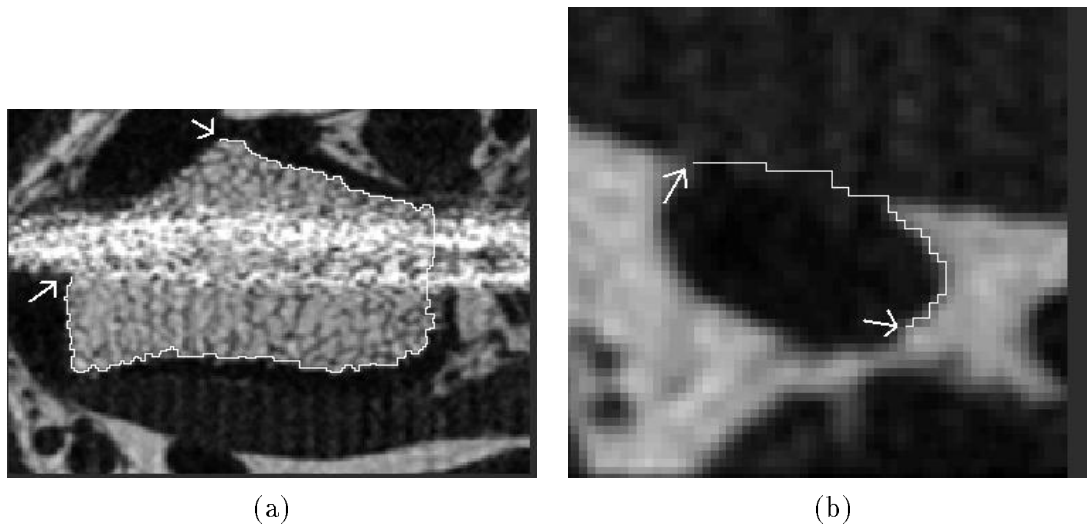


Figura 3.4: (a) Fatia da junta do pulso obtida por MRI. O segmento de *live-wire* mostra-se imune ao tipo de ruído apresentado na borda em segmentação. (b) Fatia de um veia do pulso obtida por MRI. O segmento de *live-wire* mostra-se imune ao buraco apresentado na borda da veia.

Na seção 2.1.1 foram descritas características independentes da orientação da borda e características sensíveis à orientação da borda. Na prática, adotar uma orientação para a borda e usar características sensíveis a esta orientação representa uma estratégia bastante eficiente para a extração de bordas em imagens digitais. O exemplo da figura 3.5 ilustra esta eficiência na mesma fatia do pé obtida por MRI e apresentada no exemplo da figura 3.1, onde a borda de interesse é a borda do osso tálus. A figura 3.5b mostra um segmento de *live-wire* obtido utilizando apenas características independentes da orientação. A figura 3.5c mostra um segmento de *live-wire* obtido entre os mesmos vértices indicados na figura 3.5b, mas usando características sensíveis à orientação. Observe dentro da região de interesse indicada nestas figuras que na figura 3.5c, diferente da figura 3.5b, o segmento de *live-wire* não é atraído pelo segmento espúrio de contraste similar ao da borda desejada. Isto porque as orientações deste segmento e da borda são opostas e esta informação foi embutida na geração de custos durante a classificação (da mesma forma que na figura 2.5). Usando características sensíveis a orientação, o usuário logo percebe antes de selecionar o primeiro segmento de borda que o traçado do *live-wire* favorece apenas uma orientação. A figura 3.5d ilustra esta observação para o mesmo vértice inicial da figura 3.5c. O segmento de *live-wire* apresentado adere a várias outras partes da cena evitando a borda desejada. Movendo o cursor no sentido da orientação da borda o usuário logo percebe que o segmento de *live-wire* adere à borda desejada como indicado na figura 3.5c. Em geral, o método *live-wire* não só requer menos

vértices como termina mais rápido a extração da borda quando utiliza características sensíveis à orientação.

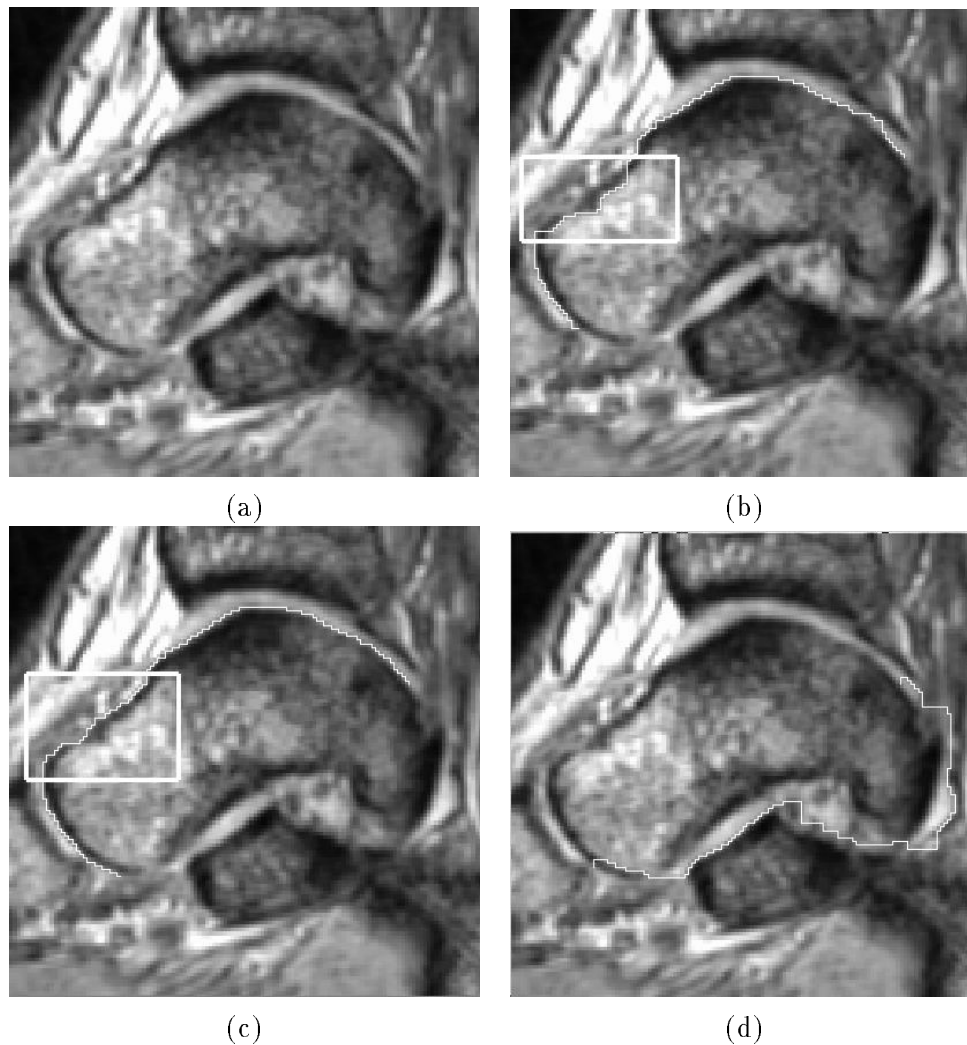


Figura 3.5: A figura ilustra a importância das características sensíveis à orientação nos métodos *live-wire*. (a) Uma fatia do pé de um paciente obtida por MRI, onde objeto desejado é o osso tálus. (b) Um segmento de *live-wire* é obtido incluindo apenas características independentes da orientação da borda. O segmento é atraído por partes espúrias de contraste similar na imagem. (c) A orientação da borda é assumida no sentido anti-horário. Um segmento de *live-wire* entre os mesmos dois vértices de (b) é obtido usando características sensíveis à orientação. A figura mostra uma região de interesse onde o segmento de contraste similar não atrai o *live-wire* visto que sua orientação não é favorável. (d) Um segmento de *live-wire* partindo do mesmo vértice inicial de (c), mas com orientação no sentido contrário, ilustra a sensibilidade do método à orientação previamente adotada para a borda.

Existem algumas variações do *live-wire* que podem ser úteis em algumas aplicações. Estas variações são discutidas nas próximas três seções e consistem respectivamente da utilização de outras funções de custo total para o caminho ótimo (seção 3.1.2), da utilização de um valor de limiar L sobre o custo acumulado durante a conectividade (seção 3.1.3) e da utilização de características estruturais durante a segmentação (seção 3.1.4).

3.1.2 Outras Funções de Custo

Existem várias maneiras de definir o custo de um caminho partindo de v_s para v_e no grafo G . Nos métodos *live*, o custo de um caminho é igual a soma do custo conjunto $c(b)$ de cada bel b deste caminho. Uma possível variação é dividir esta soma pelo comprimento (ou número de bels) do caminho. Esta normalização resulta em segmentos de *live-wire* mais longos entre vértices consecutivos reduzindo a tendência dos algoritmos de conectividade escolherem caminhos mais curtos em G . Uma outra variação de efeito similar é considerar a soma do custo $c(b)/d(b)$ onde $d(b)$ é a distância Euclideana entre o vértice final do bel b e o vértice inicial v_s . A figura 3.6 ilustra o efeito desta idéia usando a mesma fatia com os mesmos parâmetros de classificação da figura 2.5. O objeto de interesse é um osso da junta de um pulso em uma cena obtida por MRI. A figura 3.6a mostra o primeiro segmento de *live-wire* utilizando como função de custo apenas a soma do custo $c(b)$ de cada bel b deste segmento. A figura 3.6b mostra o primeiro segmento de *live-wire* partindo do mesmo vértice inicial da figura 3.6a, mas usando como função de custo a soma do custo $c(b)/d(b)$ de cada bel b . O segmento obtido nesta figura é sem dúvida muito maior do que o da figura 3.6a.

O efeito descrito acima também pode ser indesejável se a cena contém outras bordas com propriedades similares a da borda desejada. A figura 3.7 ilustra o que ocorre com a borda do tálus em uma fatia de MRI. A figura 3.7a mostra um segmento da borda de interesse obtido sem a modificação da função de custo. A figura 3.7b mostra o segmento obtido partindo do mesmo ponto inicial da figura 3.7a. Note que o segmento de *live-wire* prefere um caminho bem mais longo, mas adere a outras bordas da cena.

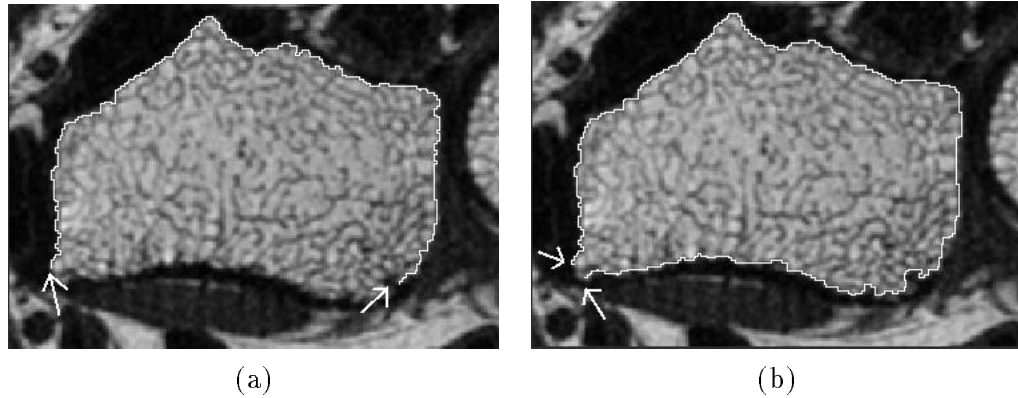


Figura 3.6: Fatia da junta do pulso obtida por MRI. A figura ilustra a utilização de outras funções de custo para o caminho ótimo. (a) Segmento de *live-wire* utilizando como função de custo a soma do custo $c(b)$ de cada bel b deste segmento. (b) Segmento de *live-wire* partindo do mesmo vértice inicial v_s de (a) usando como função de custo a soma do custo $c(b)/d(b)$ de cada bel b , onde $d(b)$ é a distância Euclideana entre o vértice final de b e v_s . Note que o segmento em (b) é bem mais longo que em (a).

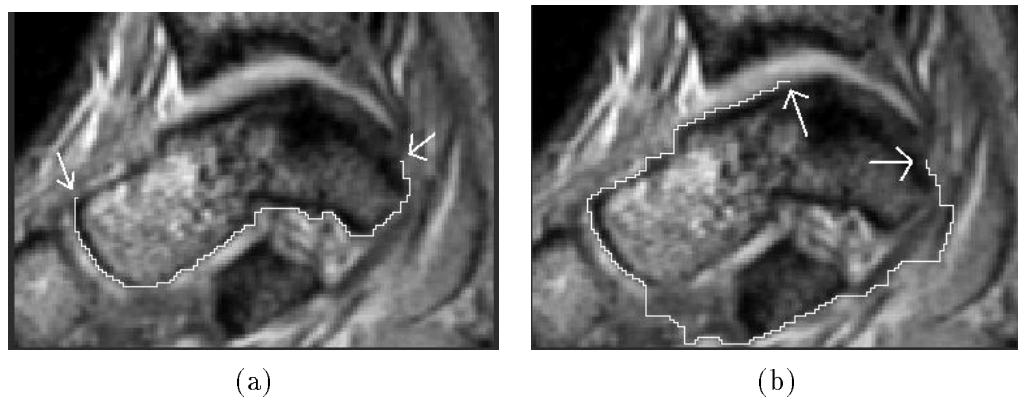


Figura 3.7: Fatia do pé obtida por MRI. A figura ilustra o efeito da função de custo $c(b)/d(b)$ na borda do osso tálus. (a) Segmento de *live-wire* desejado e obtido com a função de custo $c(b)$. (b) Segmento de *live-wire* obtido com a função de custo $c(b)/d(b)$. Note que o segmento em (b) é mais longo, mas adere a outras bordas da cena.

Portanto, seja qual for a função de custo definida para um caminho no grafo, ela deve ser uma função não decrescente com o comprimento do caminho.

3.1.3 Valor de Limiar

A idéia de usar um valor de limiar L aplicado ao custo acumulado $cc(v)$ foi discutida nas seções 2.2.1 e 2.2.2. O objetivo da idéia é reduzir o tempo de processamento dos algoritmos de conectividade. Partindo de um vértice inicial v_s sobre a borda do objeto de interesse, um caminho ótimo global é garantido para todos os vértices finais v_e cujo custo acumulado $cc(v_e)$ é menor do que L . Para que seja garantido um caminho ótimo global para todos os possíveis vértices v_e sobre a borda do objeto, L deve ser escolhido suficientemente maior do que o pior custo total possível para o contorno que descreve a borda. Como a classificação não é perfeita, o custo deste contorno é muito elevado e portanto escolher L maior do que este custo não resulta em nenhuma economia de tempo. Por outro lado, valores mais baixos de L fazem com que o comprimento dos segmentos de *live-wire* que descrevem a borda seja reduzido. Isto porque sendo o custo total de um caminho em G uma função não decrescente do seu comprimento, os vértices v_e sobre a borda do objeto que estão muito longes de v_s podem ter custos acumulados maior que L . A figura 3.8a mostra o maior segmento de *live-wire* possível que descreve a borda desejada partindo do mesmo vértice inicial v_s da figura 3.6. A figura 3.8b mostra o segundo segmento de *live-wire* seguindo a sequência de segmentos para completar a borda. Neste exemplo são necessários 11 vértices para completar a borda desejada. No entanto, o tempo de processamento para cada vértice é 25 vezes menor comparado com o tempo do algoritmo LW_2 original. Isto significa que mesmo selecionando mais vértices sobre a borda, se o tempo de processamento para cada um destes vértices é desprezível, o tempo total do usuário para completar a borda é reduzido. Para a borda da figura 3.6, apenas dois vértices são necessários usando o algoritmo LW_2 original. Se L é o tempo de processamento deste algoritmo para cada vértice, o tempo total de processamento é $2t$. Usando o valor de limiar L , o tempo total de processamento passa a ser $11t/25$ representando uma redução de mais de 4 vezes no tempo total do usuário.

Portanto, a escolha de L passa a ser um problema de otimização. O valor de limiar L deve ser escolhido tal que (i) agilize o processo evitando o cálculo desnecessário do caminho ótimo para vértices distantes da borda e (ii) reduza o número de segmentos de *live-wire* necessários para completar a borda desejada. Este valor "ótimo" de L é obtido experimentalmente para uma dada aplicação.

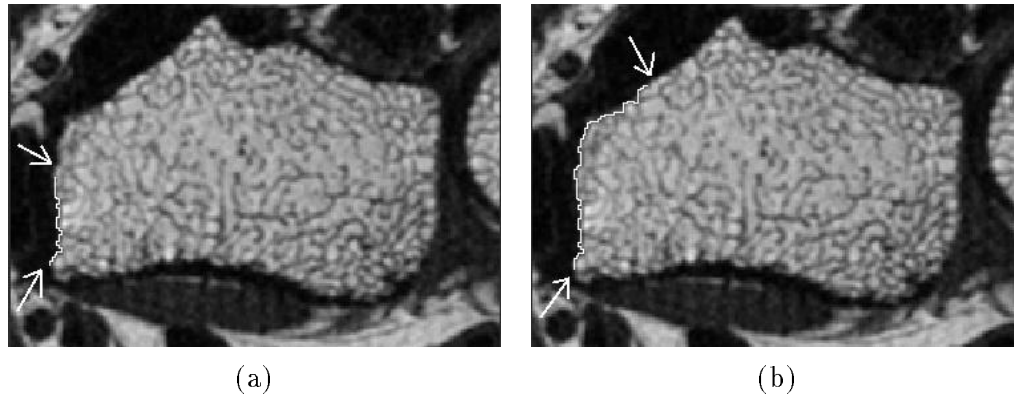


Figura 3.8: Fatia da junta do pulso obtida por MRI. A figura ilustra o efeito da utilização de um valor de limiar T nos algoritmos de conectividade. (a) Primeiro segmento de *live-wire*. (b) Segundo segmento de *live-wire*. Os segmentos ficam mais curtos e neste caso são necessários 11 segmentos para completar a borda. Em contra-partida, o tempo de processamento para cada vértice reduz 25 vezes.

3.1.4 Região Anular

As características de imagem usadas na classificação (seção 2.1) podem ser vistas como **características estáticas**. Elas são calculadas uma única vez e embutidas na função de custo conjunto que é fixa durante todo o processo dinâmico descrito pelo *live-wire*. Durante a segmentação, o formato do objeto vai sendo delineado fatia por fatia. Isto sugere a utilização de **características estruturais** na etapa de conectividade que operam como **características dinâmicas**. A questão é como utilizar características dinâmicas sem que restrições sejam impostas ao tipo de borda e sem comprometer a resposta em tempo real necessária para algoritmos interativos.

Uma variação adotada pelos métodos *live* é a utilização da característica dinâmica representada por uma região anular em torno da borda segmentada na fatia anterior e projetada na fatia atual (figura 3.9). A região anular é uma faixa de largura W centrada em torno do contorno fechado, conectado e orientado da fatia anterior. Esta região limita a área de busca dos algoritmos de conectividade reduzindo o tempo de processamento e eliminando outras bordas da cena que podem ser atrativas para o *live-wire*. A região anular se baseia em duas propriedades: (i) a segmentação é feita fatia por fatia e (ii) a forma da borda do objeto varia normalmente muito pouco de uma fatia para a outra. Para qualquer objeto com superfície digital 3D fechada e conectada é possível achar o menor valor de W tal que a borda do objeto nas fatias esteja sempre dentro da região anular. Nos métodos *live*, o usuário

escolhe o valor de W experimentalmente.

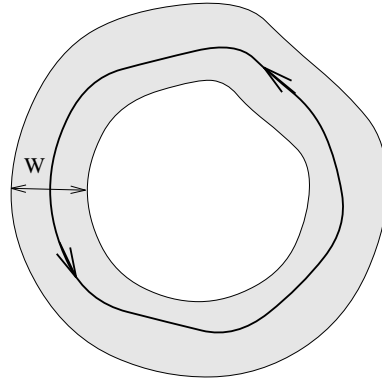


Figura 3.9: Região anular de largura W da fatia anterior projetada na fatia atual. O contorno na fatia atual está dentro da região anular.

A idéia da região anular falha quando o objeto de interesse é representado por apenas um contorno na fatia anterior e na fatia atual existem dois contornos representando este objeto (figura 3.10a). Neste caso, o usuário não usa a região anular na fatia atual, mas reativa esta característica para a próxima fatia. No caso contrário, quando dois contornos da fatia anterior se unem em um único contorno na fatia atual, o valor de W pode ser definido para que a região acompanhe esta mudança topológica (figura 3.10b). Em geral, a região anular é uma boa característica que agiliza muito a segmentação restringindo os processos de classificação e conectividade para uma faixa de pixels de largura W em vez de usar todos os pixels da cena.

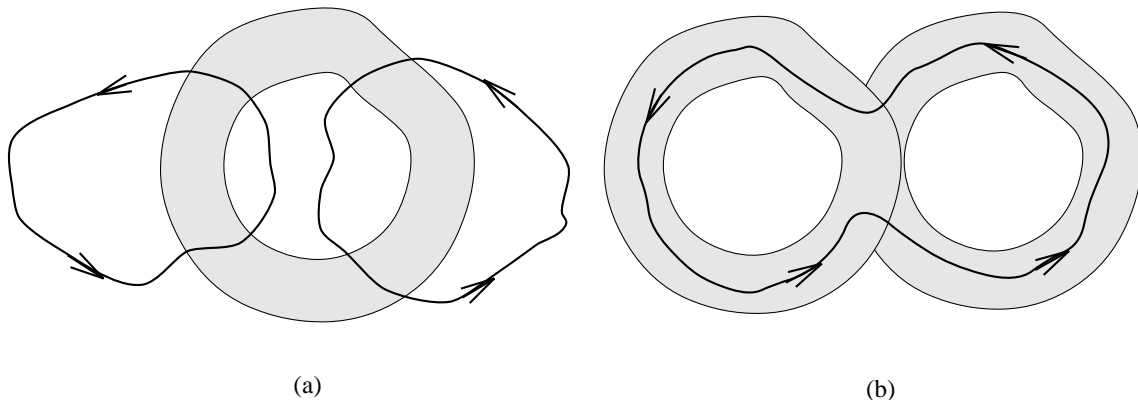


Figura 3.10: (a) Dois contornos da fatia atual não podem ser extraídos usando a região anular resultante de um único contorno da fatia anterior. (b) Duas regiões anulares resultantes de dois contornos na fatia anterior são projetadas na fatia atual. O contorno da fatia atual está totalmente inserido na região anular.

3.2 Live-Lane

No método *live-lane*, dado uma cena $\mathbf{C} = (C, g)$, o usuário inicia o processo selecionando o vértice v_0 sobre a borda de interesse e em seguida conduz o cursor do *mouse* na vizinhança da borda dentro de uma faixa de pixels conforme ilustrado na figura 3.11. Esta faixa de pixels é criada de tal forma que para qualquer instante t existe uma matriz quadrada de pixels $S(t) \subset C$ onde o vértice indicado pela posição do cursor pertence a um pixel de $S(t)$. Os vértices $v_i, i = 1, 2, \dots, n$, de V escolhidos sobre a borda do objeto são selecionados intermitentemente e automaticamente durante o processo. Eventualmente, nada impede que o usuário também selecione um vértice v_i como em *live-wire*. Segmentos de *live-wire* confinados em $S(t)$ são calculados entre cada par de vértices $(v_i, v_{i+1}), i = 0, 1, \dots, n - 1$, e apresentados em tempo real na tela do computador. Quando, após percorrer a borda, o cursor retorna à vizinhança de v_0 , o usuário indica uma operação de "fechamento" para iniciar o cálculo do último segmento de *live-wire* entre o vértice mais recente, v_n (posição atual do cursor), e o vértice v_0 . Assim *live-lane* é um processo mais dinâmico que difere de *live-wire* nos seguintes fatores: (i) o mecanismo pelo qual os vértices v_i de V são escolhidos (ii) a extensão na imagem do grafo direcionado usado para encontrar os caminhos ótimos entre os vértices v_i e v_{i+1} de V e (iii) o processo de interação homem-máquina. Abaixo são descritas duas estratégias para *live-lane* que influenciam de forma diferente estes três fatores.

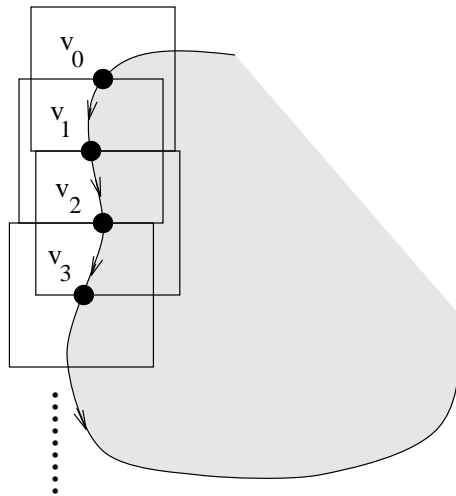


Figura 3.11: A figura ilustra o processo de segmentação usando *live-lane*.

Seja $T = (t_0, t_1, \dots, t_n)$ uma sequência de instantes de tempo representando a discretização do intervalo de tempo $[t_0, t_n]$, onde t_0 e t_n indicam respectivamente os instantes de tempo para o início e fim do processo *live-lane* em \mathbf{C} . T representa uma sequência de

instantes de tempo para os quais as posições do cursor são lidas durante o processo. Para qualquer t_i em T , $v_i \in V$ e $S(t_i)$ denotam respectivamente o vértice do pixel em C apontado pelo cursor e a matriz quadrada de pixels no instante t_i . Um vértice v é dito pertencer a $S(t_i)$ se v é um vértice de um pixel em $S(t_i)$. O número de colunas de $S(t_i)$ define a área quadrada de pixels em \mathbf{C} onde há a busca por caminhos ótimos e é chamado **largura da faixa**.

3.2.1 Faixa de Largura Fixa

Para qualquer t_i em T , a matriz quadrada $S(t_i)$ define uma **subcena** $\mathbf{S}(t_i) = (S(t_i), g_S(t_i))$ em \mathbf{C} , onde $g_S(t_i)$ é a restrição de g para $S(t_i)$. Neste método assume-se que a largura de faixa F para $S(t_i)$ é fixa para todos t_i ¹.

Após seleção do primeiro vértice v_0 em \mathbf{C} no instante t_0 , $S(t_0)$ é definida como sendo a matriz quadrada de pixels centrada em v_0 . Enquanto o usuário move o cursor dentro de $S(t_0)$, cada posição do cursor v define um vértice em $S(t_0)$ para o qual um caminho ótimo partindo de v_0 é calculado e apresentado em tempo real usando um dos algoritmos de conectividade. Quando o cursor cruza $S(t_0)$, o caminho ótimo calculado em $\mathbf{S}(t_0)$ de v_0 para o último vértice v_1 indicado em $S(t_0)$ é selecionado automaticamente como um segmento de borda². Este v_1 passa agora a ser o próximo vértice inicial e se t_1 é o instante de tempo para o qual v_1 foi selecionado, então v_1 passa a ser o centro de $S(t_1)$ e $\mathbf{S}(t_1)$ é a próxima subcena para cálculo de caminho ótimo. O processo continua até algum t_n , quando o primeiro vértice v_0 selecionado cai dentro da subcena $\mathbf{S}(t_n)$ e o usuário indica a operação de "fechamento". Neste instante, o último segmento de borda é calculado de v_n a v_0 e um contorno fechado, conectado e orientado é obtido.

Existem várias vantagens da abordagem *live-lane* sobre o método *live-wire*. Primeiro, o método *live-wire* pode não encontrar velocidade adequada de resposta em computadores menos potentes (a menos que use o valor de limiar L discutido na seção 3.1.3 e/ou a região anular discutida na seção 3.1.4). Especialmente se a borda desejada ocupar uma grande área de pixels na cena. Em contraste, *live-lane* oferece resposta em tempo real em qualquer computador, incluindo PCs mais antigos, visto que a busca por caminhos ótimos está confinada ao grafo associado com $\mathbf{S}(t_i)$, o qual é significativamente menor do que o grafo G associado com a cena \mathbf{C} . Isto significa que em *live-lane* a redução do tempo do usuário

¹Na implementação de *live-lane* a largura F é um parâmetro livre que pode ser selecionado no intervalo de 5 a 100 pixels.

²Na prática v_1 pode não estar exatamente sobre a borda e por isso v_1 é escolhido como o vértice do caminho ótimo que está a 70% de seu comprimento.

para segmentar a cena é mais ou menos independente da potência do computador. Segundo, quando a borda tem muitos segmentos mal definidos, o usuário precisa selecionar vértices mais próximos e portanto muito mais segmentos de *live-wire* são necessários para extrair a borda. Em *live-lane*, isto ocorre naturalmente para pequenos valores de F e, portanto, a velocidade do método não é tão afetada pela má qualidade da borda como em *live-wire*. Para $F = 1$, por exemplo, *live-lane* se torna um traçado manual.

A discussão acima leva à seguinte questão: Por quê não variar F automaticamente durante o processo, para que maiores valores de F sejam usados em regiões bem definidas da borda e valores menores em regiões mal definidas? A resposta a esta pergunta é o assunto da próxima seção.

3.2.2 Faixa de Largura Variável

Nesta estratégia, uma decisão crucial a ser feita é como determinar as partes mal e bem definidas da borda do objeto durante o traçado. Qualquer método computacional automático (tal como aqueles baseados na intensidade de gradientes) se depara com o dilema de ter que conhecer a região da cena na qual está sendo feito o cálculo, o que traz à tona o problema básico da segmentação de imagens; o reconhecimento automático do objeto. A solução explora mais uma vez a superioridade do usuário em reconhecer o objeto comparado com os algoritmos de computador. Durante o traçado, o usuário naturalmente reduz a velocidade do cursor para delinear as partes mal definidas da borda do objeto e aumenta esta velocidade nas partes bem definidas. A velocidade e a aceleração do cursor são, portanto, usadas para determinar a largura de $S(t_i)$.

Para qualquer $t_i \in T$, v_i é um vetor com duas componentes indicando as duas coordenadas do vértice em um sistema de coordenadas Cartesianas. A velocidade $s(t_i)$ e a aceleração $a(t_i)$ são definidas como

$$\begin{aligned} s(t_0) &= a(t_0) = 0, \\ s(t_i) &= \frac{\|v_i - v_{i-1}\|}{t_i - t_{i-1}}, \quad i > 0, \\ a(t_i) &= \frac{s(t_i) - s(t_{i-1})}{t_i - t_{i-1}}, \quad i > 0, \end{aligned} \tag{3.1}$$

onde $\|\cdot\|$ denota magnitude vetorial. Seja $lw(t_i)$ o número de colunas da matriz $S(t_i)$ o qual é chamado **largura da faixa** no tempo t_i . Esta estratégia difere da anterior em dois

aspectos: (i) $lw(t_i)$ pode variar com t_i e (ii) vértices v_i são selecionados definindo segmentos de borda ótimos todas as vezes que $lw(t_i)$ variar.

O processo começa no instante em que usuário seleciona o vértice inicial v_0 com um pequeno valor para $lw(t_0)$ (e.g. 10 pixels). Enquanto o usuário move o cursor, um caminho ótimo para cada vértice v apontado pelo cursor dentro de $S(t_0)$ é calculado e apresentado em tempo real. Para $t_0 \leq t \leq t_1$, $lw(t) = lw(t_0)$ (i.e., $S(t) = S(t_0)$). Durante o traçado, os vértices v_i , $i = 1, 2, \dots, n$, vão sendo selecionados automaticamente cada vez que um dos dois seguintes eventos ocorre: (i) o cursor cruza a borda de $S(t_{i-1})$ ou (ii) $|a(t)| > A$, onde $t \geq t_{i-1}$ e A é um valor fixo de limiar. Para este instante $t = t_i$, $lw(t_i)$ é definido como

$$lw(t_i) = Ks(t_i), \quad (3.2)$$

onde $K > 0$ é uma constante e $S(t_i)$ é centrada em v_i com largura $lw(t_i)$. O caminho ótimo de v_{i-1} a v_i é selecionado como segmento de borda e v_i passa a ser o novo vértice inicial. O processo continua até que para algum instante de tempo t_n o usuário indique uma operação de "fechamento" e v_0 esteja dentro de $S(t_n)$. Neste instante, o caminho ótimo entre v_n e v_0 é calculado e apresentado como o último segmento de borda finalizando o processo.

Embora a largura da faixa seja calculada como uma função linear da velocidade do cursor, a aceleração também é necessária para interpretar as ações do usuário durante o processo. Como as transições entre regiões bem definidas e regiões mal definidas (ou vice-versa) são abruptas ao longo da borda, altos valores positivos (ou negativos) de aceleração são lidos nestas regiões. Isto indica para o algoritmo que a posição atual do cursor deve ser selecionada como o novo vértice inicial v_i e a largura da faixa deve ser atualizada mesmo se o vértice ainda estiver dentro de $S(t_{i-1})$. Com isto, o usuário passa a guiar o cursor com alta velocidade e baixa aceleração em regiões bem definidas, com baixa velocidade e baixa aceleração em regiões mal definidas e com alta aceleração/desaceleração em regiões de transição.

A eficácia relativa de *live-wire* e *live-lane* é estudada em detalhes na próxima seção.

3.3 Avaliação

Métodos interativos de segmentação de imagens são normalmente utilizados em situações onde os métodos automáticos falham e o traçado manual é a única alternativa imediata. Portanto, a validação de um método interativo deve em primeiro lugar avaliar sua

eficiência comparando-o com o traçado manual. O objetivo desta seção é obter uma avaliação objetiva e quantitativa da utilidade dos métodos *live-wire* e *live-lane*, usando como base de comparação o traçado manual.

Existem muitos parâmetros envolvidos nos métodos *live-wire* e *live-lane*: as características de imagem, suas transformadas e parâmetros, e os pesos da equação 2.18; a largura W da região anular, o valor de limiar L , e a função de custo de um caminho ótimo para o método *live-wire*; a largura da faixa F para o método *live-lane* com faixa de largura fixa; os valores de K , A e o grau de resposta interativa aos movimentos do cursor para o método *live-lane* com faixa de largura variável. A tarefa de comparar ambos métodos levando em conta todos estes parâmetros é absurda. A abordagem adotada, portanto, elimina todos aqueles parâmetros que, através de experimentos de observação, são improváveis de comprometer a análise de performance dos métodos. Por exemplo, a função de custo conjunto (seção 2.1.2) resultante da classificação é a mesma para todos os métodos. Por este processo chega-se às seguintes versões para comparação:

- LW: *live-wire* com largura de região anular $W = 20$ pixels.
- LL: *live-lane* com faixa de largura fixa $F = 60$ pixels.
- AL: *live-lane* com faixa de largura variável onde $K = 5$ e $A = 9$.
- MN: traçado manual.

Os valores dos parâmetros acima são encontrados para uma aplicação particular da qual vem os dados de teste. O algoritmo de conectividade LW_2 (seção 2.2.2) foi implementado após a realização dos experimentos deste trabalho e portanto, apesar de LW_2 ser 2 a 3 vezes mais rápido do que LW_1 (seção 2.2.1), os experimentos foram realizados usando LW_1 . Todos os métodos são implementados em uma versão interna do *software 3DVIEWNIX* [83] e executados em uma *workstation SGI INDY*.

A utilidade dos métodos acima é medida baseada na velocidade e repetibilidade (seção 1.3.2). A exatidão dos métodos é assumida decorrente da aceitação de um usuário especialista que supervisiona o resultado dos experimentos, visto que determinar a "verdade absoluta" em imagens médicas é uma tarefa muito complexa. Apesar do traçado manual ser considerado uma boa aproximação da "verdade absoluta" em alguns trabalhos, o uso desta técnica nos métodos *live* tornou-se questionável visto que a repetibilidade deste método é pior do que a repetibilidade dos paradigmas *live*. O que só vem a confirmar a motivação apresentada na seção 1.1 sobre a superioridade do operador humano (com relação aos algo-

ritmos de computador) em reconhecer o objeto na cena e a superioridade dos algoritmos de computador (comparados com o operador humano) em delinear o objeto na cena.

A velocidade e repetibilidade dos métodos acima são medidas em dados de uma aplicação para análise de cinemática de juntas tarsais do pé [82]. Nesta aplicação, os movimentos tridimensionais de juntas tarsais do pé são estudados *in vivo* para obterem medidas quantitativas e qualitativas dos movimentos normais e anormais. A meta a longo prazo é o planejamento cirúrgico levando em conta o movimento das juntas. Para tanto, mais de 10,000 cenas 2D foram adquiridas via MRI constituindo dados de mais de 20 juntas de vivos, cada qual em 8 posições diferentes. A maior parte das idéias nos métodos *live* vêm do esforço para desenvolver métodos práticos e efetivos de segmentar os ossos destas juntas. A dificuldade desta segmentação resulta da falta de informação da ressonância magnética para ossos, a presença de múltiplas bordas de osso adjuntas e aos tendões e ligamentos que se prendem aos ossos. Nesta aplicação, os métodos *live* têm sido utilizados para definir a borda 4D de 4 ossos do pé: tálus, calcâneo, navicular e cubóide. Os experimentos de avaliação foram realizados utilizando o tálus e o calcâneo. A figura 3.12 mostra uma fatia do pé de um paciente obtida por MRI indicando estes ossos. A razão para considerar o tálus e o calcâneo é que eles representam condições bem diferentes para a tarefa de segmentação. A borda do tálus consiste basicamente de segmentos bem definidos com pequenas partes mal definidas. Em contraste, a borda do calcâneo possui muitos segmentos longos e mal definidos.



Figura 3.12: Uma fatia do pé de um paciente obtida por MRI mostrando as bordas dos ossos tálus e calcâneo usadas na avaliação dos métodos *live*.

Um conjunto Σ de 30 cenas 2D é selecionado para avaliar os métodos. Cada um de três operadores humanos O_1, O_2 e O_3 extrai destas cenas em três tentativas separadas, denotadas E_1, E_2, E_3 , o tálus (T_a) e o calcâneo (C_a) usando cada um dos quatro métodos LW, LL, AL e MN. O contorno resultante da segmentação de cada cena 2D $\mathbf{C}=(C, g) \in \Sigma$ é convertido em uma cena binária $\mathbf{C}_e = (C_e, g_e)$, onde $C_e = C$ e para qualquer $p \in C$, $g_e(p) = 1$ para p no interior do contorno e $g_e(p) = 0$ para p no seu exterior. Note que uma cena binária resultante de um experimento contém informação sobre apenas um osso. Para quaisquer $o \in \{O_1, O_2, O_3\}$, $b \in \{T_a, C_a\}$, $r \in \{E_1, E_2, E_3\}$, $m \in \{LW, LL, AL, MN\}$ e $\mathbf{C} \in \Sigma$, o ato do operador o de traçar o contorno do osso b usando o método m na tentativa r é referido como um **experimento de segmentação** envolvendo o, r, m, b e \mathbf{C} . Este estudo de avaliação consiste portanto de 2.160 experimentos de segmentação.

A **velocidade de segmentação** SP_e de qualquer experimento e é expressa em número de cenas \mathbf{C} (fatias) por minuto e é definida como

$$SP_e = \frac{1}{T_e}, \quad (3.3)$$

onde T_e é o tempo em minutos gasto para completar e . Considere quaisquer $o \in \{O_1, O_2, O_3\}$, $r \in \{E_1, E_2, E_3\}$, $m \in \{LW, LL, AL, MN\}$ e $b \in \{T_a, C_a\}$. A **velocidade de segmentação** SP_{ormb} do operador o na tentativa r usando o método m para extrair o objeto b é definida como a média de todas as velocidades SP_e sobre todos os experimentos de segmentação e envolvendo o, r, m, b e todas cenas 2D. A **velocidade de segmentação** SP_{omb} do operador o usando o método m para extrair o objeto b é definida como a média de todas as velocidades SP_e sobre todos os experimentos e envolvendo o, m, b e todas as tentativas e cenas 2D.

A tabela 3.1 lista os valores de SP_{ormb} para todos possíveis valores de o, r, m e b . A tabela 3.2 lista as diferenças de velocidade entre o método MN e cada um dos métodos LW, LL e AL para cada operador o e objeto b . A segunda (parte inferior) entrada nesta tabela indica a significância estatística (o valor *prob*) das diferenças de velocidade obtidas pelo “teste t do estudante” [59]. A tabela 3.3 mostra os valores de SP_{omb} para todos o, m , and b .

		T_a			C_a		
		E_1	E_2	E_3	E_1	E_2	E_3
LW	O_1	7.51	5.87	6.17	5.41	5.73	5.32
	O_2	2.90	3.28	3.34	4.19	3.93	3.14
	O_3	5.50	5.65	6.66	5.96	6.18	6.16
LL	O_1	5.94	10.28	7.72	7.69	7.19	7.96
	O_2	3.87	4.19	3.18	3.68	3.69	4.30
	O_3	6.32	7.70	7.37	7.96	6.98	7.83
AL	O_1	8.21	7.65	8.93	7.11	7.31	7.31
	O_2	2.52	3.35	3.19	3.11	3.40	3.99
	O_3	7.11	7.53	7.98	8.75	7.70	8.61
MN	O_1	2.97	3.07	2.78	3.86	3.44	3.37
	O_2	3.50	4.19	3.39	4.04	3.71	5.40
	O_3	3.44	4.22	3.74	4.35	4.77	5.09

Tabela 3.1: Velocidade de segmentação (fatias/minuto) SP_{orb} para todos possíveis valores de o , r , m , e b .

	T_a			C_a		
	LW,MN	LL,MN	AL,MN	LW,MN	LL,MN	AL,MN
O_1	3.51	4.66	5.29	1.94	4.06	3.70
	0.000	0.000	0.000	0.000	0.000	0.000
O_2	0.50	0.03	0.68	0.58	0.41	0.81
	0.111	0.914	0.024	0.119	0.279	0.024
O_3	2.12	3.30	3.75	1.39	2.85	3.61
	0.000	0.000	0.000	0.001	0.000	0.000

Tabela 3.2: Diferença em velocidade $|SP_{orb} - SP_{om'b}|$ (parte superior) e seu nível de significância estatística (parte inferior) para todos operadores o , objetos b e $m' = MN$ e $m \in \{LW, LL, AL\}$.

		SP_{omb}	
		T_a	C_a
LW	O_1	6.52	5.49
	O_2	3.17	3.75
	O_3	5.94	6.10
LL	O_1	7.98	7.61
	O_2	3.75	3.89
	O_3	7.13	7.59
AL	O_1	8.26	7.24
	O_2	3.02	3.50
	O_3	7.54	8.35
MN	O_1	2.94	3.56
	O_2	3.69	4.39
	O_3	3.80	4.74

Tabela 3.3: Velocidade de segmentação SP_{omb} (fatias/minuto) para operador, método e objeto.

Seja \mathbf{C}_e uma cena binária resultante de um experimento de segmentação e . A notação $|\mathbf{C}_e|$ é usada para denotar o número de pixels de \mathbf{C}_e com valor 1. A operação OU-exclusivo XOR entre duas cenas binárias $\mathbf{C}_{e_1} = (C_{e_1}, g_{e_1})$ e $\mathbf{C}_{e_2} = (C_{e_2}, g_{e_2})$ tal que $C_{e_1} = C_{e_2}$, escrita como $\mathbf{C}_{e_1} XOR \mathbf{C}_{e_2}$, resulta em uma cena binária $\mathbf{C}_e = (C_e, g_e)$ onde $C_e = C_{e_1} = C_{e_2}$, e, para qualquer $p \in C_e$, $g_e(p) = 1$ se $g_{e_1}(p) \neq g_{e_2}(p)$ e $g_e(p) = 0$ se $g_{e_1}(p) = g_{e_2}(p)$. Sejam e_1 e e_2 quaisquer experimentos de segmentação envolvendo o mesmo objeto $b \in \{T_a, C_a\}$ e cena $\mathbf{C} \in \Sigma$ e sejam \mathbf{C}_{e_1} e \mathbf{C}_{e_2} as cenas binárias resultante destes experimentos respectivamente. A **repetibilidade de segmentação**, $RP_{e_1 e_2}$, para o objeto b da cena \mathbf{C} nos experiments e_1 e e_2 é definida como

$$RP_{e_1 e_2} = 1 - \frac{|\mathbf{C}_{e_1} XOR \mathbf{C}_{e_2}|}{|\mathbf{C}_{e_1}| + |\mathbf{C}_{e_2}|}. \quad (3.4)$$

Para quaisquer $o_1 \in \{O_1, O_2, O_3\}$, $o_2 \in \{O_1, O_2, O_3\}$, $r_1 \in \{E_1, E_2, E_3\}$, $r_2 \in \{E_1, E_2, E_3\}$, $m \in \{LW, LL, AL, MN\}$ e $b \in \{T_a, C_a\}$, a **repetibilidade de segmentação** $RP_{o_1 o_2 r_1 r_2 m b}$ dos operadores o_1 e o_2 nas tentativas r_1 e r_2 usando o método m para extrair o objeto b é definida como a média de todas $RP_{e_1 e_2}$ para todos experimentos de segmentação e_1 e e_2 tais que $e_1 \neq e_2$, e_1 e e_2 são conduzidos na mesma cena 2D de entrada, e_1 envolve o operador o_1 e a tentativa r_1 , e_2 envolve o operador o_2 e tentativa r_2 , e ambos envolvem o método m e o objeto b . $RP_{o_1 o_2 r_1 r_2 m b}$ mede a repetibilidade *inter*-operador quando $o_1 \neq o_2$ e mede a repetibilidade *intra*-operador quando $o_1 = o_2$. A **repetibilidade de segmentação**

$RP_{o_1 o_2 m b}$ dos operadores o_1 e o_2 usando o método m para extrair o objeto b é definida como a média das repetibilidades $RP_{o_1 o_2 r_1 r_2 m b}$ sobre todas combinações de tentativas r_1 e r_2 . Finalmente, a repetibilidade de segmentação $RP_{m b}$ usando o método m para extrair o objeto b é a média de todas $RP_{o_1 o_2 r_1 r_2 m b}$ sobre todas combinações de tentativas r_1 e r_2 e operadores o_1 e o_2 e método m e objeto b .

Na tabela 3.4 são listadas as repetibilidades $RP_{o o r_1 r_2 m b}$ para todos possíveis valores de o , r_1 , r_2 , m e b . A tabela 3.5 mostra as diferenças em repetibilidade de segmentação entre MN e cada um dos métodos LW, LL e AL para cada operador e objeto. A segunda (parte inferior) entrada desta tabela indica a significância estatística das diferenças em repetibilidade obtidas pelo “teste t do estudante” [59]. A tabela 3.6 mostra as repetibilidades de segmentação $RP_{o_1 o_2 m b}$ para todos os valores de o_1 , o_2 , m e b . Finalmente, a tabela 3.7 mostra os valores de $RP_{m b}$ para todos m e b .

		T_a			C_a		
		E_1, E_2	E_2, E_3	E_1, E_3	E_1, E_2	E_2, E_3	E_1, E_3
LW	O_1	0.993	0.994	0.992	0.984	0.986	0.981
	O_2	0.992	0.992	0.991	0.985	0.966	0.966
	O_3	0.995	0.994	0.995	0.993	0.992	0.993
LL	O_1	0.992	0.995	0.991	0.966	0.970	0.982
	O_2	0.992	0.990	0.990	0.979	0.971	0.969
	O_3	0.994	0.993	0.993	0.981	0.984	0.986
AL	O_1	0.990	0.993	0.992	0.970	0.970	0.978
	O_2	0.987	0.985	0.987	0.977	0.974	0.973
	O_3	0.990	0.993	0.991	0.970	0.978	0.967
MN	O_1	0.977	0.978	0.977	0.972	0.974	0.972
	O_2	0.970	0.965	0.967	0.966	0.959	0.958
	O_3	0.975	0.978	0.975	0.968	0.969	0.966

Tabela 3.4: Repetibilidade de segmentação $RP_{o o r_1 r_2 m b}$ para todos possíveis valores de o , r_1 , r_2 , m , e b .

	T_a			C_a		
	LW,MN	LL,MN	AL,MN	LW,MN	LL,MN	AL,MN
O_1	0.016	0.015	0.014	0.011	0.000	0.000
	0.000	0.000	0.000	0.038	0.935	0.995
O_2	0.025	0.023	0.019	0.011	0.012	0.014
	0.000	0.000	0.000	0.083	0.036	0.008
O_3	0.019	0.017	0.015	0.025	0.016	0.004
	0.000	0.000	0.000	0.000	0.003	0.588

Tabela 3.5: Diferença em repetibilidade $|RP_{oomb} - RP_{oom'b}|$ (parte superior) e seu nível de significância estatística (parte inferior) para cada operador o , objeto b e $m' = MN$ e $m \in \{LW, LL, AL\}$.

		T_a			C_a		
		O_1	O_2	O_3	O_1	O_2	O_3
LW	O_1	0.993	0.989	0.987	0.984	0.974	0.975
	O_2	0.989	0.992	0.984	0.974	0.972	0.966
	O_3	0.987	0.984	0.995	0.975	0.966	0.993
LL	O_1	0.993	0.987	0.986	0.973	0.967	0.965
	O_2	0.987	0.990	0.980	0.967	0.973	0.965
	O_3	0.986	0.980	0.993	0.965	0.965	0.984
AL	O_1	0.992	0.984	0.985	0.973	0.965	0.963
	O_2	0.984	0.986	0.977	0.965	0.975	0.958
	O_3	0.985	0.977	0.991	0.963	0.958	0.971
MN	O_1	0.977	0.961	0.967	0.973	0.947	0.957
	O_2	0.961	0.967	0.959	0.947	0.961	0.944
	O_3	0.967	0.959	0.976	0.957	0.944	0.968

Tabela 3.6: Lista de repetibilidade de segmentação $RP_{o_1o_2mb}$ por operadores (o_1, o_2) , método (m) e objeto (b) .

	RP_{mb}	
	T_a	C_a
LW	0.989	0.975
LL	0.987	0.969
AL	0.985	0.966
MN	0.966	0.955

Tabela 3.7: Lista de repetibilidade de segmentação RP_{mb} por método (m) e objeto (b).

As seguintes observações resultam da análise dos resultados apresentados nas tabelas 3.1 até 3.7.

Existe uma variação considerável de velocidade entre os operadores para quaisquer método e objeto. As velocidades de O_2 diferem com significância estatística das obtidas por O_1 e O_3 . Para O_2 , parece não existir nenhuma vantagem em velocidade para os métodos *live* comparada com MN. Ao contrário, para ambos O_1 e O_3 , todos os métodos *live* oferecem um aumento de velocidade, com significância estatística ($prob < 0.0005$), com relação ao traçado manual. O aumento de velocidade encontrado por O_1 e O_3 varia entre 1,5 a 3 vezes mais rápido. Para estes operadores, LL e AL parecem ser mais rápidos do que LW. Existe variação de velocidade entre tentativas para um dado operador, método e objeto. Isto talvez reflita o aspecto de aprendizado que é inerente neste experimento. Para minimizar este efeito são tomadas duas precauções. Primeiro, um anatomista experiente revê cada fatia mostrando aos operadores a forma das bordas e todas suas localizações. Segundo, a sequência de experimentos é embaralhada de modo que nunca o mesmo objeto é extraído em sucessivos experimentos (E_1, E_2, E_3 não indica a ordem na qual as tentativas foram realizadas). O aprendizado é um fenômeno importante e difícil de ser tratado objetivamente. Fica difícil saber qual a influência do aprendizado e da experiência nas velocidades relativas dos métodos.

Repetibilidade para os métodos *live* é geralmente melhor do que para o traçado manual. Repetibilidade *intra*-operador é geralmente melhor do que repetibilidade *inter*-operador para todos operadores, métodos e objetos. Repetibilidade é melhor para T_a (pois a borda é melhor definida) do que para C_a . Para T_a , o aumento de repetibilidade (1,5%-2,5%) apresenta significância estatística para todos operadores e métodos comparados com o traçado manual. O aumento não é tão evidente para o objeto de borda mal definida C_a . Repetibilidade decresce de LW para LL para AL para MN na ordem em que os métodos se tornam menos automáticos (tabela 3.7).

3.4 Conclusão

Neste capítulo são apresentados dois paradigmas de segmentação guiada pelo usuário, *live-wire* e *live-lane*. Os métodos têm por objetivo reduzir o tempo gasto pelo usuário durante o processo. Em *live-wire* apenas poucos vértices (2 a 5) são necessários para completar a segmentação da borda do objeto em uma cena 2D. Em situações onde um número muito maior de vértices são necessários para segmentação, o método *live-lane* é utilizado.

Nota-se que à medida que os usuários vão se familiarizando com os métodos, eles conseguem terminar a segmentação em menos tempo. Quando o número de cenas é muito grande, algum tempo gasto no ajuste fino dos parâmetros da classificação e na estratégia de operação dos algoritmos pode representar uma boa economia de tempo do usuário no resto do processo.

Existem alguns aspectos da interação homem-máquina os quais são difíceis de serem quantificados e influenciam na utilidade dos métodos. O aprendizado é um deles. Um outro fator importante é a fadiga. O método LL parece causar menos fadiga em tarefas de segmentação extensas do que os outros métodos.

Capítulo 4

3D-Live-Wire

Nos métodos *live*, dado um conjunto $V = \{v_0, v_1, \dots, v_n\}$ de $n + 1$ vértices ordenados sobre a borda do objeto em \mathbf{C} , a borda completa do objeto resulta de n segmentos de *live-wire* do tipo $\langle v_i, v_{i+1} \rangle$, $i = 0, 1, \dots, n - 1$, mais o segmento $\langle v_n, v_0 \rangle$. No método *live-wire*, o usuário seleciona cada vértice v_i manualmente durante o traçado orientado da borda em \mathbf{C} . No método *live-lane*, o usuário seleciona o vértice inicial v_0 e subsequentemente guia o cursor em torno da borda no sentido de sua orientação para que os outros vértices v_i sejam selecionados automaticamente (ou eventualmente pelo usuário). Nestes dois métodos, o usuário tem que estar presente assistindo o processo de detecção de borda fatia por fatia. No método *3D-live-wire*, a segmentação é feita para cada cena 3D (ou subcena 3D de uma cena 4D) isoladamente. Dada uma cena 3D, a assistência do usuário é requerida inicialmente para selecionar cortes ortogonais ao plano das fatias e para traçar a borda do objeto nas imagens destes cortes usando *live-wire*¹. Em seguida, para todas as fatias \mathbf{C} da cena, os vértices v_i são selecionados e ordenados automaticamente sobre a borda do objeto que, em seguida, é delineada automaticamente fatia por fatia.

Neste capítulo, o método *3D-live-wire* é apresentado e avaliado. A apresentação do método (seção 4.1) consiste basicamente do mecanismo de interação com o usuário, dos processos de seleção e ordenação de vértices e da detecção de borda. Na seção 4.2, o método *live-lane* é escolhido, baseado na preferência dos usuários, como base de comparação com o *3D-live-wire*. Vários experimentos são realizados comparando estes dois métodos. A seção 4.3 conclui este capítulo e discute algumas operações de processamento de imagens que podem ser úteis para melhorar o desempenho do *3D-live-wire*.

¹Nada impede que o método *live-lane* seja utilizado nesta etapa no lugar do *live-wire*.

4.1 Metodologia

Uma cena 3D, denotada $\mathbf{C}^{(3)}$, é um par $(C^{(3)}, g)$ que consiste de uma matriz 3D $C^{(3)}$ finita de voxels chamada **domínio da cena** e de uma função $g(v) : C^{(3)} \rightarrow [L, H]$ chamada **intensidade da cena** que associa a cada voxel v em $C^{(3)}$ um valor de intensidade dentro do intervalo $[L, H]$. Em um sistema de coordenadas Cartesianas xyz , $\mathbf{C}^{(3)}$ pode ser vista como um conjunto finito $\{\mathbf{C}_0, \mathbf{C}_1, \dots, \mathbf{C}_m\}$ de $m + 1$ fatias (ou subcenas 2D) paralelas ao plano xy , consecutivas e igualmente espaçadas ao longo do eixo z (figura 1.2).

Um objeto 3D em $\mathbf{C}^{(3)}$ é composto por uma única região de voxels conexos. Nas fatias \mathbf{C}_j , $j = 0, 1, \dots, m$, uma borda de objeto 3D pode ser representada por um ou mais contornos fechados, conectados e orientados. A informação da mudança topológica da borda ao longo do eixo z é obtida subdividindo $\mathbf{C}^{(3)}$ em intervalos I de fatias consecutivas, nos quais a borda tem topologia constante. Estes intervalos são chamados *slabs*. A figura 4.1 mostra um objeto 3D que define dois *slabs* I_1 e I_2 em $\mathbf{C}^{(3)}$. O *slab* I_1 vai da fatia \mathbf{C}_0 à \mathbf{C}_p , $p < m$, e o *slab* I_2 vai da fatia \mathbf{C}_{p+1} à \mathbf{C}_m . Em I_1 , a borda é representada por dois contornos fechados, conectados e orientados por fatia e em I_2 a borda é representada por um único contorno por fatia. Note que na subcena 3D representada pelo *slab* I_1 , o objeto 3D é representado por duas subregiões de voxels conexos que estão desconectadas entre si e que em I_2 o objeto 3D é representado por uma única subregião de voxels conexos. Estas subregiões são denominadas **estruturas** S do objeto. No caso da figura 4.1, o objeto 3D possui três estruturas S_1, S_2 e S_3 . As estruturas S_1 e S_2 pertencem ao *slab* I_1 e a estrutura S_3 pertence ao *slab* I_2 .

A estratégia básica usada no método *3D-live-wire* separa $\mathbf{C}^{(3)}$ em *slabs* I e resolve a segmentação de cada estrutura S do objeto de interesse separadamente. Note que a intersecção de qualquer plano de corte em xyz e $\mathbf{C}^{(3)}$ define uma subcena 2D. As fatias \mathbf{C}_j , $j = 0, 1, \dots, m$, são um caso particular desta regra. A motivação principal do *3D-live-wire* é reduzir o tempo total do usuário que só precisa segmentar a borda de S usando *live-wire* em um número k de imagens de corte bem menor que o número de fatias de I . Por exemplo, os pontos de um contorno que representa a borda de S em uma imagem de corte ortogonal ao plano das fatias (figura 4.2) gera pelo menos dois vértices v_0 e v_1 sobre a borda em todas as fatias de I . Nestas fatias, os segmentos de *live-wire* $\langle v_0, v_1 \rangle$ e $\langle v_1, v_0 \rangle$ definem um contorno fechado, conectado e orientado que pode ser encontrado automaticamente fatia por fatia.

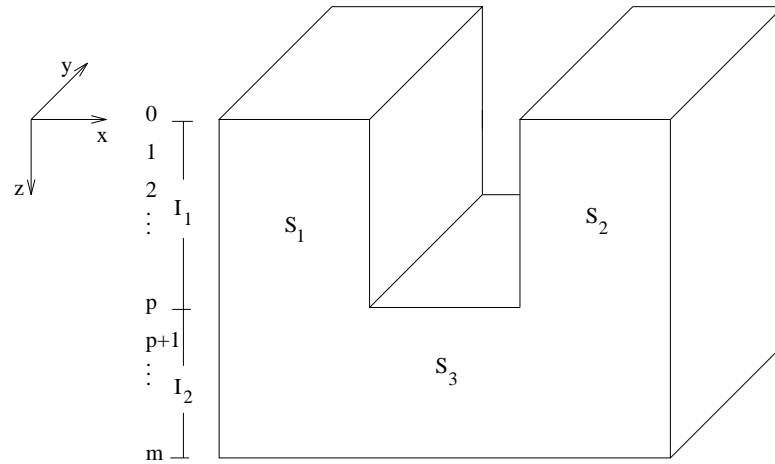


Figura 4.1: A figura mostra um objeto 3D, o qual é representado por duas estruturas (dois contornos no plano das fatias) no intervalo que vai da fatia 0 à fatia p e por uma única estrutura (um único contorno no plano das fatias) no intervalo que vai da fatia p à fatia n . Estes dois intervalos definem dois *slabs* de topologia constante para o objeto 3D.

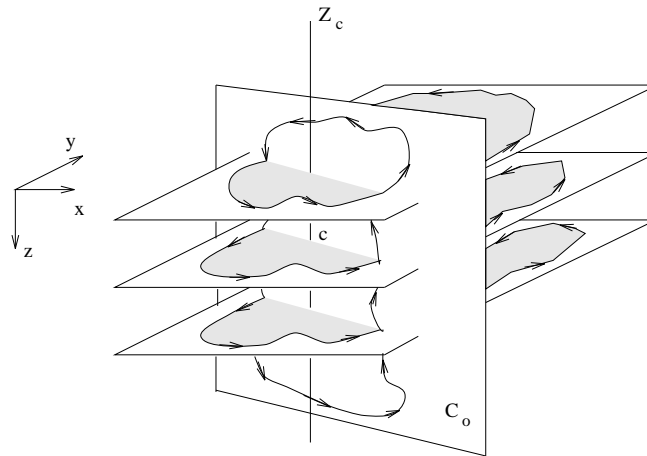


Figura 4.2: A figura mostra que um corte ortogonal ao plano das fatias de uma cena 3D. O contorno que representa a borda do objeto na cena 2D resultante do corte ortogonal gera pontos sobre a borda do objeto nas fatias da cena 3D.

O processo de segmentação é portanto dividido em cinco etapas: (i) **definição e seleção de *slabs***, (ii) **seleção de estruturas**, (iii) **seleção de cortes**, (iv) **ordenação de vértices** e (v) **detecção de borda**. As etapas (i) e (ii) funcionam da seguinte forma. Na etapa (i), o usuário visualiza na tela do computador as fatias da cena lado a lado e marca as fatias que definem o início e o fim de cada *slab* em $\mathbf{C}^{(3)}$. Os *slabs* são contínuos e,

portanto, se a fatia inicial de um *slab* I_k for \mathbf{C}_j , é porque a fatia final do *slab* anterior I_{k-1} é \mathbf{C}_{j-1} . Após subdividir $\mathbf{C}^{(3)}$ em *slabs*, o usuário escolhe um *slab* I de $\mathbf{C}^{(3)}$ que contém uma estrutura de interesse S . Esta escolha é feita selecionando qualquer fatia em I . Na etapa (ii), o usuário especifica um ponto c no interior da estrutura S em uma das fatias de I . As etapas (iii) e (iv) são descritas em detalhes nas seções 4.1.1 e 4.1.2 respectivamente. Em (iii) e (iv), um número suficiente de vértices sobre a borda da estrutura S deve ser gerado para que a detecção da borda de S nas fatias de I possa ser feita automaticamente na etapa (v) (seção 4.1.3). Neste instante, o usuário pode optar por acompanhar o processo de detecção da borda de S ou pode repetir as etapas (ii), (iii) e (iv) para outras estruturas de I (ou de outro *slab* de $\mathbf{C}^{(3)}$). A princípio, nada impede que o usuário prossiga com as etapas (ii), (iii) e (iv) para outras estruturas enquanto a etapa (v) está sendo executada para S . Na prática, no entanto, é interessante que o processo de detecção de borda seja acompanhado fatia por fatia até que o usuário se familiarize com o método. Em vista de qualquer erro, o usuário pode: (a) interromper a etapa (v) e repetir o processo a partir da fatia em que iniciou o erro ou (b) deixar chegar ao final da etapa (v) e corrigir o erro da segmentação em algumas poucas fatias isoladamente. No caso de (b), a correção pode ser feita pintando/apagando regiões de pixel da fatia que representam o interior/exterior do objeto e/ou executando *live-wire* nesta fatia. A opção atualmente utilizada é a de executar inicialmente as etapas (ii), (iii) e (iv) para todas as estruturas que compõem o objeto de interesse na cena e por fim acompanhar a detecção de todas elas simultaneamente.

4.1.1 Seleção de Cortes

Especificados um *slab* I , uma estrutura S e um ponto c marcado no interior de S em uma dada fatia \mathbf{C} do *slab* I , o usuário pode selecionar qualquer corte ortogonal ao plano das fatias (plano xy) passando por c (figura 4.2). Note que c define um eixo Z_c em xyz paralelo ao eixo z e a intersecção de todos os possíveis cortes ortogonais que passam por c deve ser o eixo Z_c . Esta restrição é fundamental para garantir o sucesso da etapa de ordenação de vértices assim como é descrita na seção 4.1.2. O usuário pode selecionar vários cortes ortogonais. Cada corte define uma subcena 2D, denotada \mathbf{C}_o , na qual o usuário traça a borda de S usando *live-wire*. A intersecção dos contornos traçados em todas subcenas \mathbf{C}_o com uma fatia \mathbf{C} de I gera um conjunto de vértices V sobre a borda da estrutura em \mathbf{C} . Estes vértices devem ser ordenados como descrito na seção 4.1.2. Cada contorno ortogonal pode gerar dois ou mais vértices em \mathbf{C} . Para facilitar o processo de ordenação, apenas os dois vértices mais distantes (vértices extremos) da intersecção de cada contorno com \mathbf{C} são selecionados.

A figura 4.3 ilustra um exemplo da seleção de cortes ortogonais na segmentação do patela indicado na figura 2.1. Este osso é representado por uma única estrutura S e um único *slab* I em $\mathbf{C}^{(3)}$. A figura 4.3a apresenta a mesma fatia do joelho da figura 1.1 usada para indicar o patela em uma cena 2D. Na figura 4.3a são indicados o ponto central c escolhido no interior do patela e dois cortes, 1 e 2, ortogonais ao plano das fatias. Estes cortes são representados por duas linhas na figura passando por c . A figura 4.3b mostra a subcena ortogonal \mathbf{C}_{o_1} resultante do corte 1 e a figura 4.3c mostra a subcena ortogonal \mathbf{C}_{o_2} resultante do corte 2. Estas figuras indicam dois traçados de *live-wire* que representam a borda do patela em \mathbf{C}_{o_1} e \mathbf{C}_{o_2} respectivamente. Note que cada linha horizontal em \mathbf{C}_{o_k} , $k = 1, 2$, representa uma fatia de $\mathbf{C}^{(3)}$ e que as duas linhas indicadas representam respectivamente as fatias inicial e final do *slab* I . Estas duas linhas guiam o usuário na localização da estrutura S em \mathbf{C}_{o_k} . No traçado indicado em \mathbf{C}_{o_k} , apenas os dois pontos de contorno dos extremos esquerdo e direito de cada linha são selecionados como vértices de V na fatia \mathbf{C} representada por esta linha. Vértices do contorno ortogonal que por ventura possam estar fora do *slab* indicado não são selecionados como vértices da borda de S em nenhuma fatia de $\mathbf{C}^{(3)}$. Esta situação é melhor ilustrada no exemplo da figura 4.4 a seguir.

A figura 4.4 mostra o processo de seleção de cortes ortogonais em uma cena 3D da junta tarsal do pé obtida por MRI. O objeto de interesse é o osso calcâneo indicado na figura 3.12. O calcâneo define em $\mathbf{C}^{(3)}$ dois *slabs*, I_1 e I_2 , e três estruturas de interesse, S_1 , S_2 e S_3 . As estruturas S_1 e S_2 pertencem ao *slab* I_1 e a estrutura S_3 pertence ao *slab* I_2 . A figura 4.4a mostra as bordas desejadas de S_1 e S_2 em uma fatia de I_1 . Os pontos c_1 e c_2 usados para selecionar respectivamente S_1 e S_2 são indicados na figura. O plano de corte passando por c_1 , que gera \mathbf{C}_{o_1} , é selecionado propositalmente como o mesmo plano de corte que passa por c_2 e gera \mathbf{C}_{o_2} . Assim $\mathbf{C}_{o_1} = \mathbf{C}_{o_2}$ e as figuras 4.4b e 4.4c mostram os traçados de *live-wire* para as bordas de S_1 em \mathbf{C}_{o_1} e S_2 em \mathbf{C}_{o_2} respectivamente. Note que o contorno ortogonal pode sair fora das linhas horizontais que marcam o *slab* I_1 , pois os pontos de contorno fora de I_1 não são selecionados como vértices da borda de nenhum S_j , $j = 1, 2, 3$, em nenhuma fatia de $\mathbf{C}^{(3)}$. A figura 4.4d mostra o mesmo plano de corte da figura 4.4a em uma fatia do *slab* I_2 . Neste caso o plano de corte passa pelo ponto c_3 , que identifica a estrutura S_3 em I_2 , e gera a subcena ortogonal $\mathbf{C}_{o_3} = \mathbf{C}_{o_1} = \mathbf{C}_{o_2}$ apresentada na figura 4.4e. A figura 4.4e mostra o traçado de *live-wire* para S_3 em \mathbf{C}_{o_3} . Note que as linhas horizontais indicam agora o *slab* I_2 e que o contorno de S_3 em \mathbf{C}_{o_3} pode sair de I_2 sem problemas.

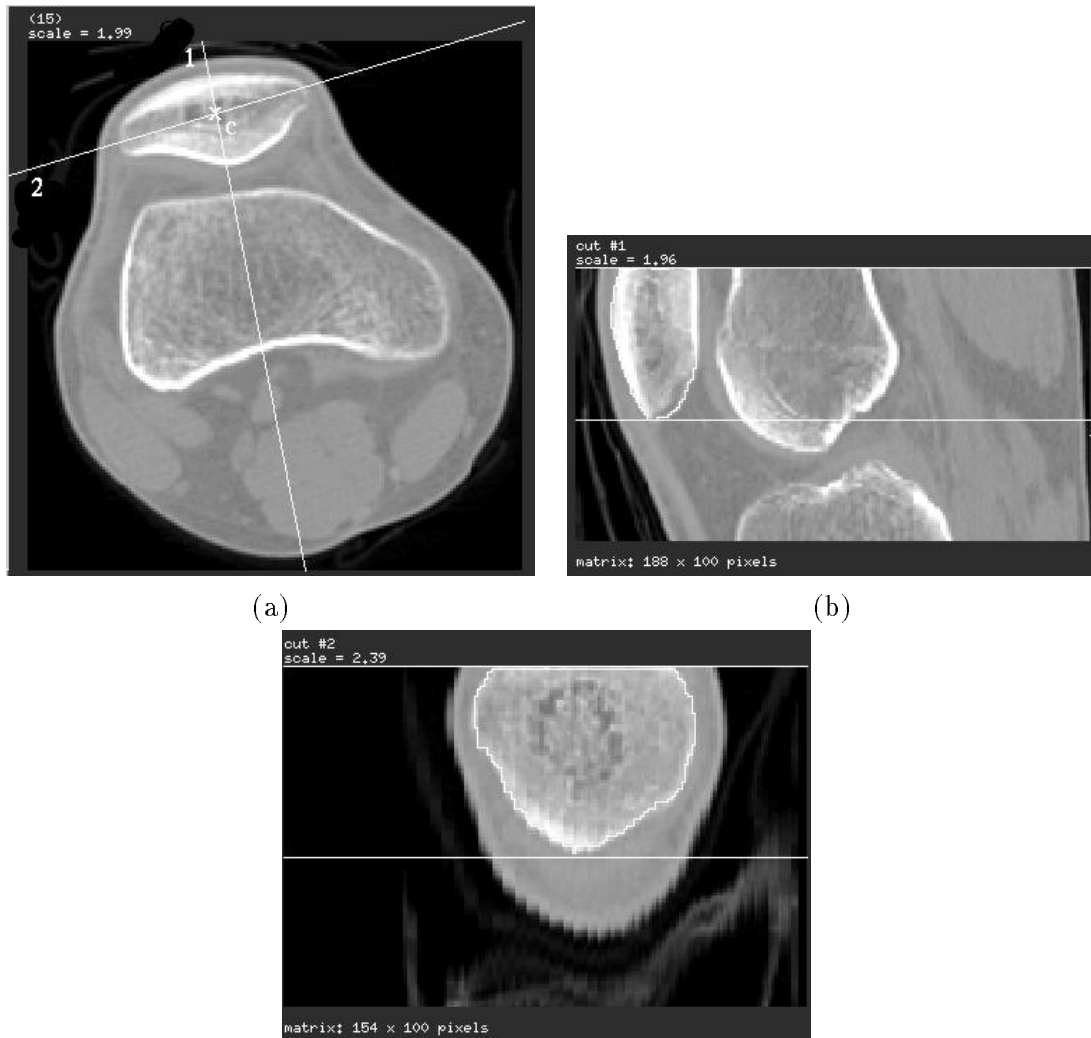


Figura 4.3: (a) Fatia de um joelho obtida por CT na qual dois cortes ortogonais C_1 e C_2 são feitos passando pelo ponto central c marcado pelo usuário no interior do patela (objeto de interesse). (b) Traçado do *live-wire* sobre a borda do patela no corte ortogonal C_1 . As duas linhas horizontais representam a primeira e a última fatia que contém este osso na cena 3D. (c) Traçado do *live-wire* sobre a borda do patela no corte ortogonal C_2 .

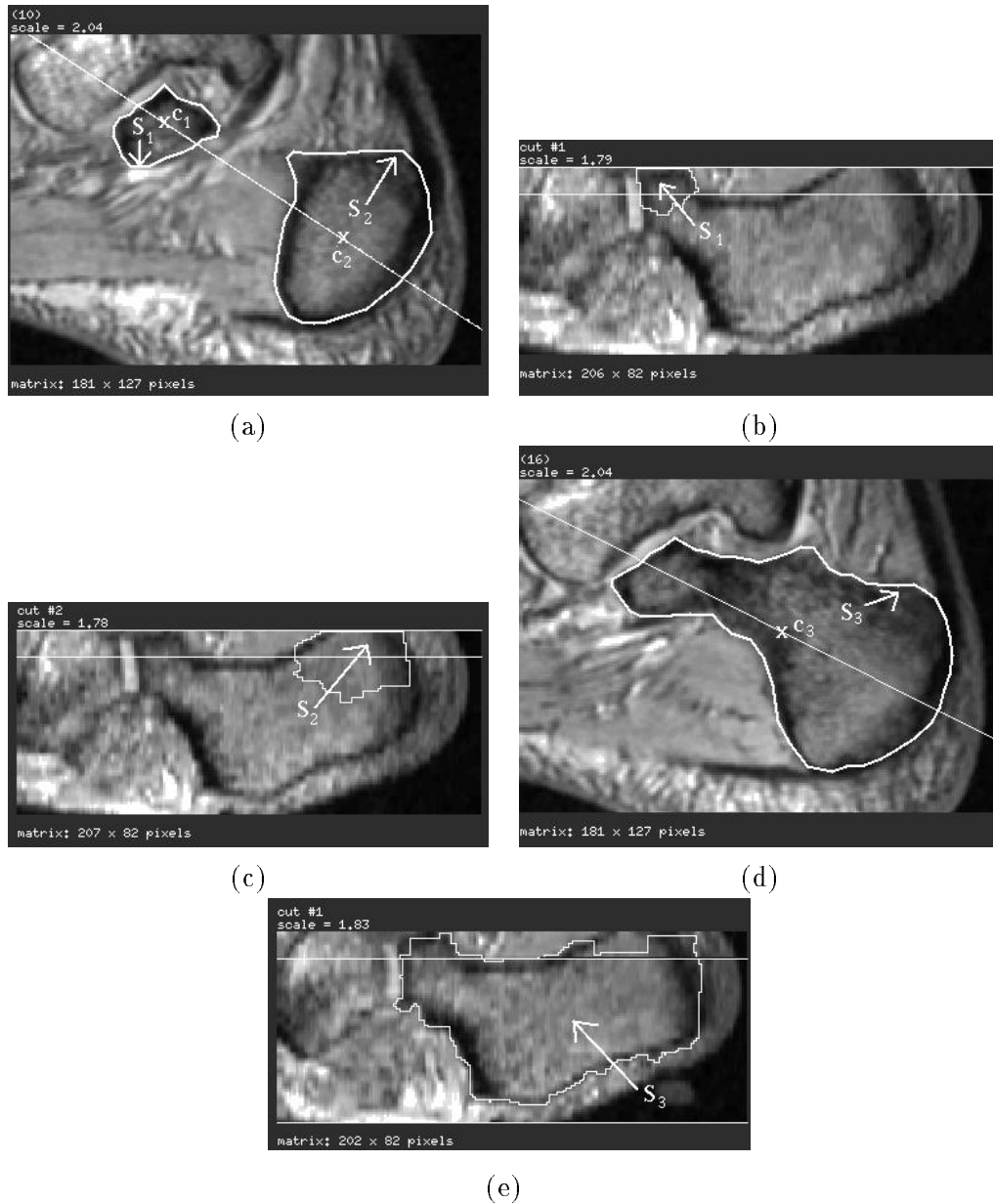


Figura 4.4: A figura ilustra a seleção de cortes ortogonais para o osso calcâneo em uma cena 3D de MRI. O calcâneo define dois *slabs*, I_1 e I_2 , e três estruturas, S_1 , S_2 e S_3 . S_1 e S_2 pertencem a I_1 e S_3 pertence a I_2 . (a) Uma fatia de I_1 indica as bordas de S_1 e S_2 . Um mesmo plano de corte passandl do por $c_1 \in S_1$ e $c_2 \in S_2$ é usado para indicar dois cortes ortogonais a S_1 e S_2 respectivamente. As figuras (b) e (c) mostram respectivamente os traçados do *live-wire* para S_1 e S_2 na imagem do corte ortogonal indicado em (a). As linhas horizontais definem as fatias inicial e final de I_1 . (d) Uma fatia de I_2 indica o mesmo plano de corte de (a) passando por $c_3 \in S_3$. (e) Imagem do corte ortogonal mostrando o traçado do *live-wire* para S_3 . As linha horizontais definem as fatias inicial e final de I_2 .

4.1.2 Ordenação de Vértices

Após a seleção de cortes para uma dada estrutura S de um dado *slab* I de $\mathbf{C}^{(3)}$, para cada fatia \mathbf{C} de I existe um conjunto V de $n + 1$ vértices sobre a borda de S em \mathbf{C} . Se k é o número de cortes ortogonais ao plano xy passando pelo ponto c que identifica S , então o conjunto V possui $n + 1 = 2k$ vértices em \mathbf{C} . O problema de encontrar o contorno ótimo global (custo mínimo) que é fechado, conectado, orientado e passa por todos os vértices de V pode requerer um tempo de processamento considerável sem nenhuma garantia de sucesso. Isto porque a insuficiência do processo de classificação na definição de bordas e a tendência dos algoritmos de conectividade de optarem por caminhos mais curtos no grafo normalmente fazem com que este contorno não corresponda à borda desejada. Como foi visto no capítulo 3, a borda desejada é na verdade ótima por partes. Portanto, para extrair esta borda é interessante que os vértices de V tenham uma **ordem de processamento**. Estes vértices devem ser ordenados conforme a orientação da borda de S em \mathbf{C} .

O processo de ordenação de vértices é descrito com o auxílio da figura 4.5. Seja c o ponto de intersecção do eixo Z_c com a fatia \mathbf{C} . O ponto c define a origem de um sistema de coordenadas Cartesianas $x'y'$ como indicado na figura 4.5. Cada corte ortogonal forma um ângulo θ com o eixo horizontal x' medido no sentido horário como indicado na figura. O ângulo θ varia de 0° a 180° representando todos os possíveis planos de corte ortogonais ao plano das fatias passando por c . Os vértices gerados em \mathbf{C} são classificados em vértices do tipo \oplus e vértices do tipo \ominus . Esta classificação leva em conta a orientação da borda. Na figura 4.5 a borda é orientada de tal forma que o interior do objeto está sempre do lado esquerdo do contorno e o exterior está sempre do lado direito do contorno em qualquer subcena 2D de $\mathbf{C}^{(3)}$. Os vértices do tipo \oplus são aqueles em que o contorno ortogonal intercepta \mathbf{C} na direção z (entrando no plano) e os vértices do tipo \ominus são aqueles em que o contorno ortogonal intercepta \mathbf{C} na direção $-z$ (saindo do plano). Na figura 4.5, o plano de corte 1 tem ângulo θ_1 e o plano de corte 2 tem ângulo θ_2 . O plano 1 gera os vértices a_1 e b_1 e o plano 2 gera os vértices a_2 e b_2 , totalizando quatro vértices para o conjunto V . Os vértices a_1 e a_2 são do tipo \oplus e os vértices b_1 e b_2 são do tipo \ominus . O processo de ordenação consiste inicialmente em associar um valor $o(v)$ a cada vértice $v \in V$ seguindo a seguinte regra: a um vértice do tipo \oplus é associado um valor $180 - \theta + 1$ e a um vértice do tipo \ominus é associado um valor $-\theta$. Os possíveis valores de $o(v)$ estão no intervalo $[-180, 181]$. Em seguida, os vértices são classificados na ordem crescente deste valor se a borda é orientada no sentido anti-horário e são classificados na ordem decrescente deste valor se a borda é orientada no sentido horário. Para o caso da figura 4.5, $o(a_1) = 180 - \theta_1 + 1$, $o(b_1) = -\theta_1$,

$o(a_2) = 180 - \theta_2 + 1$ e $o(b_2) = -\theta_2$. Como $\theta_1 < \theta_2$ e ambos estão no intervalo $[0, 180]$, a ordem crescente de valor é $-\theta_2, -\theta_1, 180 - \theta_2 + 1, 180 - \theta_1 + 1$ e os vértices passam a ter a seguinte ordem de processamento em V : b_2, b_1, a_2 e a_1 . Note que a ordem de seleção dos planos de corte não afeta o processo de ordenação. A prova de validade deste processo para um número arbitrário de cortes segue abaixo.

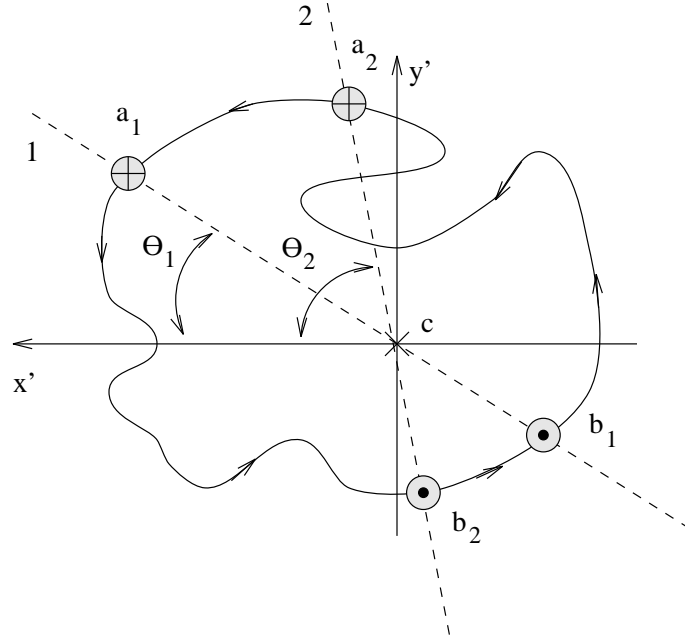


Figura 4.5: Os pontos do contorno ortogonal fechado, conectado e orientado levam as informações da orientação da borda e do ângulo de corte quando são selecionados nas fatias. Estas informações são usadas para encontrar a ordem dos pontos na fatia de acordo com a orientação da borda.

Seja $\{\theta_1, \theta_2, \dots, \theta_K\}$ o conjunto de ângulos associados a K planos de corte ortogonais ao plano das fatias passando por c . A ordem de seleção destes planos é arbitrária. Seja $\theta_k < \theta_{k+1}$ para todo $0 \leq k \leq K - 1$ onde $\theta_k \in [0, 180]$ para qualquer $0 \leq k \leq K$. Sejam a_k e b_k o par de vértices externos associados ao plano de corte de ângulo θ_k , onde $0 \leq k \leq K$, a_k é do tipo \oplus e b_k é do tipo \ominus . O objetivo é provar que o processo de ordenação sempre gera uma sequência $(b_K, b_{K-1}, \dots, b_1, a_K, a_{K-1}, \dots, a_1)$ para uma borda orientada no sentido anti-horário e gera uma sequência $(a_1, a_2, \dots, a_K, b_1, b_2, \dots, b_K)$ para uma borda no sentido horário.

O processo de ordenação associa valores $o(a_k) = 180 - \theta_k + 1$, $k = 1, 2, \dots, K$, para os vértices $a_k \in V$ e valores $o(b_k) = -\theta_k$, $k = 1, 2, \dots, K$, para os vértices $b_k \in V$. Como

$\theta_k \in [0, 180]$, então $o(a_k) \in [1, 181]$ e $o(b_k) \in [-180, 0]$ para qualquer $0 \leq k \leq K$. Isto significa que $o(a_{k_1}) > o(b_{k_2})$ para quaisquer $1 \leq k_1, k_2 \leq K$. Como $\theta_k < \theta_{k+1}$ para todo $0 \leq k \leq K-1$, então a ordem crescente de valor entre os b_k é $o(b_K) < o(b_{K-1}) < \dots < o(b_1)$ e a ordem crescente de valor entre os a_k é $o(a_K) < o(a_{K-1}) < \dots < o(a_1)$. Como $o(a_{k_1}) > o(b_{k_2})$ para quaisquer $1 \leq k_1, k_2 \leq K$, então a ordem crescente dos valores de o resulta na sequência $(b_K, b_{K-1}, \dots, b_1, a_K, a_{K-1}, \dots, a_1)$ e a ordem decrescente destes valores resulta na sequência $(a_1, a_2, \dots, a_K, b_1, b_2, \dots, b_K)$ como queríamos demonstrar.

4.1.3 Detecção de Borda

Finalizado o processo de ordenação de vértices para uma estrutura S de um *slab* I , para qualquer fatia \mathbf{C} de I existe um conjunto $V = \{v_0, v_1, \dots, v_n\}$ de $n + 1$ vértices ordenados sobre a borda de S em \mathbf{C} . No processo de detecção de borda, a borda completa de S em \mathbf{C} resulta de n segmentos de *live-wire* do tipo $\langle v_i, v_{i+1} \rangle$, $i = 0, 1, \dots, n-1$, mais o segmento $\langle v_n, v_0 \rangle$. Estes segmentos são calculados e apresentados sequencialmente em tempo real na tela do computador. Este processo se repete fatia por fatia para todas as fatias de I . O usuário pode acompanhar a detecção de borda de cada estrutura S isoladamente ou pode acompanhar a detecção de todas estruturas S de uma só vez. Nesta segunda opção, para cada fatia \mathbf{C} de $\mathbf{C}^{(3)}$, o algoritmo detecta sequencialmente todas as bordas de todas as estruturas S presentes na fatia \mathbf{C} .

Antes de iniciar a detecção de borda, o usuário pode especificar o valor W da característica de região anular descrita na seção 3.1.4. Como para cada segmento $\langle v_i, v_{i+1} \rangle$, o vértice final $v_e = v_{i+1}$ é fixo, o algoritmo de conectividade LW_2 pode parar assim que v_{i+1} sai da fila Q . Esta modificação de LW_2 foi discutida na seção 2.2.2. Ela reduz de 7 a 20 vezes o tempo de processamento do algoritmo LW_2 sendo portanto a versão atualmente utilizada no *3D-live-wire*.

4.2 Avaliação

O objetivo desta seção é obter uma avaliação objetiva e quantitativa do método *3D-live-wire*. Considerando que os métodos *live-wire* e *live-lane* são mais rápidos e mais repetitivos do que o traçado manual (seção 3.3), e que o *live-lane* teve melhor aceitação por parte dos operadores humanos envolvidos na avaliação dos métodos, o método *live-lane* é utilizado como base de comparação. Os resultados obtidos podem ser extrapolados para comparar o

3D-live-wire com o traçado manual. A abordagem adotada elimina todos parâmetros que são improváveis de comprometer a análise de desempenho entre os métodos. Por exemplo, os parâmetros de classificação geram a mesma função de custo para ambos métodos. O *3D-live-wire* é implementado em uma versão interna do *software 3DVIEWNIX* [83]. Os experimentos de avaliação usam a mesma *workstation SGI INDY*, os ossos tálus e calcâneo da mesma cena 3D usada na avaliação descrita na seção 3.3, e o algoritmo de conectividade LW_1 (seção 2.2.1). As versões utilizadas são:

- 3D: *3D-live-wire* com largura de região anular $W = 20$ pixels.
- LL: *live-lane* com faixa de largura fixa $F = 60$ pixels.

Dado um número fixo de cortes ortogonais necessários para segmentar um objeto 3D em $\mathbf{C}^{(3)}$, o desempenho do *3D-live-wire* depende do número de fatias de $\mathbf{C}^{(3)}$ nas quais este objeto está presente. Para avaliar este aspecto do *3D-live-wire* com significância estatística no mínimo 7 experimentos são necessários [8]. Estes experimentos são realizados da seguinte forma. Uma cena 3D, denotada $\mathbf{C}^{(3)}_{32}$, com 32 fatias foi selecionada e interpolada gerando o conjunto $\mathcal{C} = \{\mathbf{C}^{(3)}_{21}, \mathbf{C}^{(3)}_{27}, \mathbf{C}^{(3)}_{32}, \mathbf{C}^{(3)}_{38}, \mathbf{C}^{(3)}_{43}, \mathbf{C}^{(3)}_{50}, \mathbf{C}^{(3)}_{63}\}$ de sete cenas 3D com 21, 27, 32, 38, 43, 50 e 63 fatias respectivamente. Nestas cenas 3D, o tálus (T_a), que necessita de 6 cortes ortogonais para ser segmentado, foi escolhido como objeto de interesse. Cada um de dois operadores humanos, O_1 e O_2 extrai a borda 3D do T_a usando cada um dos dois métodos 3D e LL em cada uma das cenas de \mathcal{C} . O tempo total (em minutos) de interação de cada operador $o \in \{O_1, O_2\}$ usando cada um dos métodos $m \in \{3D, LL\}$ para extrair completamente a borda 3D do T_a em cada uma das cenas 3D de \mathcal{C} é medido e apresentado na tabela 4.1.

número de fatias	tempo do usuário (minutos)			
	LL		3D	
	O_1	O_2	O_1	O_2
21	3.75	4.50	2.17	2.42
27	4.17	5.28	2.17	2.42
32	5.82	6.35	2.17	2.42
38	6.25	7.88	2.17	2.42
43	7.67	8.67	2.17	2.42
50	8.92	10.42	2.17	2.42
63	10.17	12.00	2.17	2.42

Tabela 4.1: Tempo do usuário (minutos) para segmentar T_a para todos possíveis valores de o e m usando cenas 3D com diferente número de fatias.

Em seguida, a cena $\mathbf{C}^{(3)}_{32} = (C^{(3)}_{32}, g_{32})$ com 32 subcenas 2D é selecionada e cada um de dois operadores humanos O_1 e O_2 extrai desta cena em sete experimentos separados, denotados E_1, E_2, \dots, E_7 , o tálus (T_a) e o calcâneo (C_a) usando cada um dos dois métodos 3D e LL. O resultado da segmentação é convertido em uma cena 3D binária $\mathbf{C}^{(3)}_e = (C^{(3)}_e, g_e)$, onde $C^{(3)}_e = C^{(3)}_{32}$ e para qualquer $v \in C^{(3)}_{32}$, $g_e(v) = 1$ para v no interior do objeto e $g_e(v) = 0$ para v no seu exterior. Note que a cena binária resultante de um experimento contém informação sobre apenas um osso. Este estudo de avaliação consiste portanto de 56 experimentos de segmentação.

Considere quaisquer $o \in \{O_1, O_2\}$, $e \in \{E_1, E_2, \dots, E_7\}$, $m \in \{3D, LL\}$ e $b \in \{T_a, C_a\}$. A **velocidade de segmentação** $SP^{(3)}_{oemb}$ do operador o no experimento e usando o método m para extrair o objeto b é definida como o número de fatias dividido pelo tempo total (em minutos) de interação do usuário durante a segmentação. A **velocidade de segmentação** $SP^{(3)}_{omb}$ do operador o usando o método m para extrair o objeto b é definida como a média de todas as velocidades $SP^{(3)}_{oemb}$ sobre todos os experimentos e envolvendo o , m e b . A **velocidade de segmentação** $SP^{(3)}_{mb}$ do método m para extrair o objeto b é definida como a média de todas as velocidades $SP^{(3)}_{oemb}$ sobre todos operadores o envolvendo m e b .

A tabela 4.2 lista os valores de $SP^{(3)}_{oemb}$ para todos possíveis valores de o , e , m e b . A tabela 4.3 lista as diferenças de velocidade entre o método LL e o método 3D para cada operador o e objeto b . A segunda (parte inferior) entrada nesta tabela indica a significância estatística (o valor *prob*) das diferenças de velocidade obtidas pelo “teste t do estudante” [59]. A tabela 4.4 mostra os valores de $SP^{(3)}_{omb}$ para todos o , m , e b e a tabela 4.5 mostra os valores de $SP^{(3)}_{mb}$ para todos m e b .

		T_a		C_a	
		O_1	O_2	O_1	O_2
LL	E_1	5.36	4.55	3.73	3.92
	E_2	5.65	4.85	4.05	4.09
	E_3	5.49	5.02	3.72	4.11
	E_4	5.41	5.11	4.28	4.21
	E_5	5.36	5.16	4.35	4.32
	E_6	5.65	5.25	4.42	4.27
	E_7	5.57	5.33	4.52	4.27
3D	E_1	13.71	11.43	6.00	4.56
	E_2	16.00	11.64	5.47	4.69
	E_3	14.75	12.80	4.73	5.30
	E_4	13.81	13.71	5.60	5.63
	E_5	14.22	14.22	5.37	5.47
	E_6	16.27	13.91	5.18	5.66
	E_7	14.44	15.00	5.35	5.79

Tabela 4.2: Velocidade de segmentação (fatias/minuto) $SP^{(3)}_{omb}$ para todos possíveis valores de o , e , m , e b , usando uma cena 3D com 32 fatias.

	T_a	C_a
	LL,3D	LL,3D
O_1	9.24	1.23
	0.000	0.008
O_2	8.21	1.13
	0.001	0.010

Tabela 4.3: Diferença em velocidade $|SP^{(3)}_{omb} - SP^{(3)}_{om'b}|$ (parte superior) e seu nível de significância estatística (parte inferior) para todos operadores o , objetos b e $m' = 3D$ e $m = LL$.

	T_a		C_a	
	O_1	O_2	O_1	O_2
LL	5.50	5.04	4.15	4.17
3D	14.74	13.24	5.39	5.30

Tabela 4.4: Velocidade de segmentação (fatias/minuto) $SP^{(3)}_{omb}$ para todos possíveis valores de o , m , e b , usando uma cena 3D com 32 fatias.

	T_a	C_a
LL	5.27	4.16
3D	13.99	5.35

Tabela 4.5: Velocidade de segmentação (fatias/minuto) $SP^{(3)}_{mb}$ para todos possíveis valores de m e b , usando uma cena 3D com 32 fatias.

Seja $\mathbf{C}^{(3)}_e$ uma cena binária resultante de um experimento de segmentação e . A notação $|\mathbf{C}^{(3)}_e|$ é usada para denotar o número de voxels de $\mathbf{C}^{(3)}_e$ com valor 1. A operação OU-exclusivo XOR entre duas cenas binárias $\mathbf{C}^{(3)}_{e_1} = (C^{(3)}_{e_1}, g_{e_1})$ e $\mathbf{C}^{(3)}_{e_2} = (C^{(3)}_{e_2}, g_{e_2})$ tal que $C^{(3)}_{e_1} = C^{(3)}_{e_2}$, escrita como $\mathbf{C}^{(3)}_{e_1} XOR \mathbf{C}^{(3)}_{e_2}$, resulta em uma cena binária $\mathbf{C}^{(3)}_e = (C^{(3)}_e, g_e)$ onde $C^{(3)}_e = C^{(3)}_{e_1} = C^{(3)}_{e_2}$, e, para qualquer $v \in C^{(3)}_e$, $g_e(v) = 1$ se $g_{e_1}(v) \neq g_{e_2}(v)$ e $g_e(v) = 0$ se $g_{e_1}(v) = g_{e_2}(v)$. Sejam e_1 e e_2 quaisquer experimentos de segmentação envolvendo o mesmo objeto $b \in \{T_a, C_a\}$ e cena $\mathbf{C}^{(3)}_{32}$ e sejam $\mathbf{C}^{(3)}_{e_1}$ e $\mathbf{C}^{(3)}_{e_2}$ as cenas binárias resultante destes experimentos respectivamente. A **repetibilidade de segmentação**, $RP^{(3)}_{e_1e_2}$, para o objeto b nos experimentos e_1 e e_2 é definida como

$$RP^{(3)}_{e_1e_2} = 1 - \frac{|\mathbf{C}^{(3)}_{e_1} XOR \mathbf{C}^{(3)}_{e_2}|}{|\mathbf{C}^{(3)}_{e_1}| + |\mathbf{C}^{(3)}_{e_2}|}. \quad (4.1)$$

Para quaisquer $o_1 \in \{O_1, O_2\}$, $o_2 \in \{O_1, O_2\}$, $e_1 \in \{E_1, E_2, \dots, E_7\}$, $e_2 \in \{E_1, E_2, \dots, E_7\}$, $m \in \{3D, LL\}$ e $b \in \{T_a, C_a\}$, a **repetibilidade de segmentação** $RP^{(3)}_{o_1o_2e_1e_2mb}$ dos operadores o_1 e o_2 nos experimentos e_1 e e_2 usando o método m para extrair o objeto b é definida como a média de todas $RP^{(3)}_{e_1e_2}$ para todos experimentos de segmentação e_1 e e_2 tais que $e_1 \neq e_2$, e_1 e e_2 , e_1 envolve o operador o_1 , e_2 envolve o operador o_2 e ambos envolvem o método m e o objeto b . $RP^{(3)}_{o_1o_2e_1e_2mb}$ mede a repetibilidade *inter*-operador quando $o_1 \neq o_2$ e mede a repetibilidade *intra*-operador quando $o_1 = o_2$. A **repetibilidade de segmentação** $RP^{(3)}_{o_1o_2mb}$ dos operadores o_1 e o_2 usando o método m para extrair o objeto b é definida como a média das repetibilidades $RP^{(3)}_{o_1o_2e_1e_2mb}$ sobre todas combinações de experimentos e_1 e e_2 . Finalmente, a **repetibilidade de segmentação** $RP^{(3)}_{mb}$ usando o método m para extrair o objeto b é a média de todas $RP^{(3)}_{o_1o_2mb}$ sobre todos operadores o_1 e o_2 e método m e objeto b .

Na tabela 4.6 são listadas as repetibilidades $RP^{(3)}_{o_1o_2mb}$ para todos possíveis valores de o_1 , o_2 , m e b . A tabela 4.7 mostra as diferenças em repetibilidade de segmentação entre LL e 3D para cada operador e objeto. A segunda (parte inferior) entrada desta tabela indica a significância estatística das diferenças em repetibilidade obtidas pelo “teste t do estudante” [59]. Finalmente, a tabela 4.8 mostra os valores de $RP^{(3)}_{mb}$ para todos m e b .

		T_a		C_a	
		O_1	O_2	O_1	O_2
LL	O_1	0.9922	0.9890	0.9845	0.9713
	O_2	0.9890	0.9909	0.9713	0.9737
3D	O_1	0.9950	0.9925	0.9814	0.9767
	O_2	0.9925	0.9948	0.9767	0.9875

Tabela 4.6: Repetibilidade $RP^{(3)}_{oo'mb}$ para todos possíveis valores de o , o' , m , e b .

	T_a	C_a
	LL,3D	LL,3D
O_1	0.0028	0.0031
	0.0000	0.0001
O_2	0.0039	0.0137
	0.0000	0.0001

Tabela 4.7: Diferença de repetibilidade $|RP^{(3)}_{omb} - RP^{(3)}_{om'b}|$ (parte superior) e seu nível de significância estatística (parte inferior) para todos operadores o , objetos b e $m' = 3D$ e $m = LL$.

	T_a	C_a
LL	0.9903	0.9752
3D	0.9937	0.9806

Tabela 4.8: Repetibilidade $RP^{(3)}_{mb}$ para todos possíveis valores de m e b .

As seguintes observações resultam da análise dos resultados apresentados nas tabelas 4.1 até 4.8.

Para segmentar o tálus, o *3D-live-wire* requer cerca de 2 minutos do tempo do usuário e é independente do número de fatias, enquanto que o tempo do usuário usando o *live-lane* aumenta com o número de fatias. Variando o número de fatias de 21-63 (tabela 4.1), o *3D-live-wire* é 2-6 vezes mais rápido que o *live-lane* e consequentemente 3-15 vezes mais rápido que o traçado manual. Em cenas com poucas fatias, o tempo do usuário usando *live-lane* pode ser menor do que usando *3D-live-wire*. Para a cena com 32 fatias, o *3D-live-wire* é 2,6 vezes mais rápido que o *live-lane* para segmentar o tálus e 1,3 vezes mais rápido para segmentar o

calcâneo. O calcâneo define dois *slabs* e três estruturas requerendo um total de 20 cortes, o que justifica a queda de velocidade do *3D-live-wire* para segmentá-lo. O *3D-live-wire* é mais repetitivo, com significância estatística, que o *live-lane* e, conseqüentemente, mais repetitivo que o traçado manual.

4.3 Conclusão

Neste capítulo é apresentado o terceiro paradigma de segmentação guiada pelo usuário, o método *3D-live-wire*. Comparado com os métodos *live-wire* e *live-lane*, o *3D-live-wire* pode reduzir ainda mais o tempo gasto pelo usuário durante o processo. Em alguns casos, porém, dependendo de uma relação K/N , onde K é o número de cortes necessários para segmentar o objeto e N é o número de fatias da cena em que o objeto está presente, os métodos *live-wire* e *live-lane* podem ser uma alternativa mais rápida do que o *3D-live-wire*. Quando a aplicação envolve a segmentação de muitas cenas, o *3D-live-wire* causa muito menos fadiga do que *live-wire* e *live-lane*.

Note que existem muitas tarefas no método *3D-live-wire* que podem ser executadas em paralelo, tais como, a detecção de múltiplas bordas em uma dada fatia da cena e a detecção de borda de uma estrutura S_1 fatia por fatia enquanto o usuário está selecionando cortes ortogonais para outra estrutura S_2 . No momento, o *3D-live-wire* usa apenas processamento sequencial.

Para obter melhor desempenho do *3D-live-wire*, o usuário pode reformatar [25] a cena 3D em alguma outra direção (e.g. sagital, coronal, etc.) visando minimizar o número de *slabs* e o número de cortes necessários para segmentar o objeto. O objeto da figura 4.1, por exemplo, define apenas um *slab* e uma estrutura para fatias no plano xz (ou yz).

Na maioria das cenas 3D, os voxels não são cúbicos (i.e. $d \neq p$ na figura 1.2) e uma imagem de corte ortogonal com pixels de dimensões $d \times p$ apresenta distorções que dificultam o reconhecimento do objeto de interesse pelo usuário. Portanto, a imagem ortogonal é interpolada [25] antes de ser apresentada na tela, para que os pixels tenham dimensões $p \times p$. Após o traçado de *live-wire*, os pontos do contorno são convertidos em vértices da borda do objeto corrigindo os efeitos da interpolação. O usuário também pode optar em trabalhar com a cena 3D previamente interpolada (i.e. voxels cúbicos).

Os parâmetros de classificação, que resultam dos segmentos de borda traçados pelo usuário em uma fatia da cena durante o treinamento (seção 2.1.3), são normalmente adequa-

dos para classificar as imagens de cortes ortogonais. Entretanto, o treinamento efetuado em imagens de cortes ortogonais sempre gera uma função de custo melhor para traçar a borda do objeto nestas imagens do que a função de custo gerada pelo treinamento na fatia.

Capítulo 5

Conclusão e Discussão

Neste trabalho são propostos e apresentados três paradigmas de segmentação de imagens interativa: *live-wire*, *live-lane* e *3D-live-wire*. Os métodos visam situações onde a segmentação automática falha requerendo exaustiva assistência do usuário. Nestas situações, os paradigmas *live* reduzem o tempo gasto pelo usuário e são mais repetitivos, com significância estatística, do que o traçado manual. Em conclusão, neste capítulo são resumidos os principais aspectos dos métodos propostos, em que situações um método é melhor que outro, os resultados obtidos, as contribuições deste trabalho e algumas idéias para trabalhos futuros.

Nos métodos *live*, as bordas de objetos 2D, 3D e 4D são delineadas fatia por fatia explorando a superior habilidade do operador humano (comparada com a dos algoritmos de computador) em reconhecer a borda desejada em uma dada fatia e a superior habilidade dos algoritmos de computador (comparada com a do operador humano) em delinear esta borda.

Em *live-wire*, para segmentar uma borda em uma dada fatia, o usuário usa o cursor para selecionar um ponto inicial sobre a borda e, para qualquer posição subsequente do cursor na fatia, o segmento ótimo (caminho de custo mínimo) que parte do ponto inicial para a posição atual do cursor é calculado e apresentado na tela do computador em tempo real. Posicionando o cursor em diferentes partes da borda desejada, o usuário verifica em tempo real se os segmentos ótimos correspondentes descrevem a borda adequadamente. Para minimizar seu tempo de envolvimento no processo, o usuário deve escolher o maior segmento ótimo que descreve a borda. Este segmento é selecionado apenas quando o usuário deposita o cursor na posição atual que passa ser a do novo ponto de partida. O processo continua desta forma até completar um contorno fechado, conectado e orientado que representa a borda desejada na fatia. Normalmente 2 a 5 pontos são necessários para completar uma borda em uma dada fatia e este processo deve ser repetido para extrair todas as outras bordas do objeto fatia

por fatia. Uma desvantagem desta abordagem, porém, é que o efeito de visualizar em tempo real qualquer segmento de borda ótimo que parte do último ponto selecionado pelo usuário, requer o cálculo prévio de todos os segmentos ótimos possíveis partindo deste ponto. Isto pode obviamente comprometer o tempo de resposta às ações do usuário nos casos de imagens com muitos pixels e computadores de baixa potência. Esta desvantagem pode ser ainda mais relevante se forem necessários muitos pontos para completar a segmentação da borda. Para superar esta dificuldade são propostas duas soluções práticas: (i) o usuário pode selecionar uma região mínima de interesse que contenha a borda desejada e (ii) o usuário pode definir uma região anular em torno da borda (seção 3.1.4), restringindo a busca por caminhos ótimos ao interior desta região. Quando estas soluções não são suficientes, o método *live-lane* deve ser utilizado.

O *live-lane* também é utilizado de acordo com a preferência dos usuários com relação a seu mecanismo de interação. Em *live-lane*, o usuário também interage fatia por fatia e seu envolvimento na segmentação de uma dada fatia é maior comparado com *live-wire*. Para segmentar uma borda em uma dada fatia usando *live-lane*, o usuário seleciona apenas o primeiro ponto sobre a borda desejada e em seguida conduz rapidamente o cursor dentro de uma certa faixa de pixels em torno da borda. Pontos são selecionados automaticamente e de forma intermitente durante o traçado conduzido pelo usuário e segmentos de *live-wire* são calculados e apresentados em tempo real entre cada dois pontos consecutivos. A faixa de pixels em torno da borda é composta por um conjunto de pequenas regiões quadradas definidas em torno dos pontos selecionados a cada instante discreto de tempo. Neste caso, o cálculo de todos possíveis segmentos ótimos partindo de um dado ponto em um dado instante está restrito à região quadrada correspondente neste instante. Este cálculo é feito em tempo real e a resposta às ações do usuário são instantâneas. Um efeito interessante notado durante os experimentos é que mesmo requerendo maior envolvimento do usuário do que o *live-wire*, o *live-lane* parece causar menos fadiga em tarefas de segmentação extensas. Mesmo assim, alguns usuários ainda preferem e/ou conseguem em alguns casos terminar a segmentação em menos tempo usando o *live-wire*. Uma desvantagem em ambos métodos, *live-wire* e *live-lane*, é que o usuário tem que estar presente interagindo na segmentação de todas as fatias da cena. Para superar esta desvantagem, o método *3D-live-wire* é proposto.

Em *3D-live-wire*, o usuário divide inicialmente a cena 3D em intervalos contínuos de fatias consecutivas, tal que o objeto 3D seja representado em cada intervalo por um número constante de bordas por fatia. Em um dado intervalo, o objeto 3D é portanto representado por um número constante de estruturas 3D que estão desconexas neste intervalo e cada estrutura define uma única borda em qualquer fatia do intervalo ou corte ortogonal ao plano das fatias

passando pela estrutura. Para cada estrutura 3D de cada intervalo de fatias, o usuário seleciona cortes ortogonais ao plano das fatias passando pela estrutura correspondente e executa o método *live-wire* para delinear a borda da estrutura em cada corte. Após selecionar cortes para todas estruturas, se estes cortes forem escolhidos estrategicamente, um número suficiente de pontos ordenados é gerado sobre cada borda de cada estrutura em cada fatia da cena e segmentos de *live-wire* são calculados entre pontos consecutivos e apresentados em tempo real na tela do computador. O usuário portanto acompanha o processo automático de segmentação de todas as bordas do objeto 3D fatia por fatia. Obviamente, a razão entre o número de cortes necessário e o número de fatias pode comprometer o desempenho do *3D-live-wire* com relação às outras abordagens 2D. Entretanto, para casos que necessitam de poucos cortes ortogonais (de 1 a 6), o *3D-live-wire* reduz consideravelmente o tempo de envolvimento do usuário no processo. Na verdade, a escolha do método mais adequado para uma dada aplicação está relacionada com a necessidade de interação do usuário com o processo. Para tanto, o usuário pode escolher um dos métodos *live* ou pode ainda combiná-los em diferentes intervalos de fatias consecutivas da cena.

A avaliação de desempenho dos métodos propostos mostra que *live-wire* e *live-lane* são de 1,5 a 2,5 vezes mais rápidos e mais repetitivos, com significância estatística, do que o traçado manual. No caso de objetos que necessitam de 6 cortes ortogonais, o *3D-live-wire* requer apenas 2 minutos de interação do usuário independente do número de fatias da cena. Variando este número de 21 a 63 fatias, conclui-se que *3D-live-wire* é cerca de 2 a 6 vezes mais rápido do que *live-lane* e conseqüentemente de 3 a 15 vezes mais rápido do que o traçado manual. Os experimentos mostram que o *3D-live-wire* é mais repetitivo, com significância estatística, do que *live-lane*. Na verdade, a repetibilidade dos métodos propostos aumenta conforme o grau de automaticidade. Considerando que os experimentos de avaliação utilizaram o algoritmo LW_1 (seção 2.2.1) e que o algoritmo LW_2 (seção 2.2.2) é 2-3 vezes mais rápido do que LW_1 , os resultados acima descritos são bem melhores quando substitui-se LW_1 por LW_2 .

Além de propor e avaliar a eficiência de três mecanismos diferentes de segmentação interativa, este trabalho também contribui mostrando que é possível resolver em tempo real o problema de detecção ótima de bordas através de busca em grafo sem usar nenhuma decisão heurística que possa comprometer o sucesso da segmentação. A eficiência da busca ótima em grafo parece não ter tido muita credibilidade no passado [3, 66]. Em parte devido a baixa potência das máquinas e a ausência de plataformas interativas com o usuário, mas por outro lado, ainda parece existir uma inércia em aceitar que em muitas situações, tentar evitar a intervenção do usuário tendo como meta a automação total do processo pode implicar em

considerável esforço de pesquisa e desenvolvimento sem produzir nenhum resultado prático no final. Outra contribuição deste trabalho está na estratégia de separar a segmentação nas etapas de classificação e conectividade. Esta estratégia é fundamental para garantir a resposta em tempo real necessária para algoritmos iterativos.

As características usadas na classificação são de fundamental importância para garantir o sucesso da conectividade. Na verdade, a maior dificuldade da segmentação de imagens é a escolha de um conjunto de características apropriado para uma dada aplicação. Um estudo mais aprofundado, portanto, sobre outras características e sobre outros mecanismos que mapeiam características em custo conjunto é recomendado. As etapas de classificação e conectividade são também independentes e podem ser combinadas com outras metodologias de segmentação. Um exemplo é a utilização da função de custo conjunto do processo de classificação como energia externa em modelos de contornos deformáveis [10, 18, 41, 44]. Os algoritmos de conectividade também podem ser utilizados para técnicas baseadas em regiões ao invés de bordas. O algoritmo LW_2 , por exemplo, pode ser adaptado para que dado um ponto semente (pixel ou voxel) no interior do objeto de interesse, o algoritmo encontre todos os outros pixels/voxels da cena cujo custo do caminho ótimo partindo do ponto dado até o elemento seja menor que um certo limiar. Se os custos de cada aresta que une estes elementos forem gerados adequadamente, o resultado é um algoritmo de crescimento de regiões no interior do objeto de interesse.

Referências Bibliográficas

- [1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, Jun 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] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [4] R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, NJ, 1962.
- [5] M. Bomans, K.H. Höhne, U. Tiede, and M. Riemer. 3D-Segmentation of MR-images of the head for 3D-display. *IEEE Transactions on Medical Imaging*, 9(2):177–183, 1990.
- [6] A.E.O. Boudraa, J.J. Mallet, J.E. Besson, S.E. Bouyoucef, and J. Champier. Left ventricle automated detection method in gated isotopic ventriculography using fuzzy clustering. *IEEE Transactions on Medical Imaging*, 12(3):451–465, Sep 1993.
- [7] M.E. Brummer, R.M. Mersereau, R.L. Eisner, and R.R.J. Lewine. Automatic detection of brain contours in MRI data sets. In *Proceedings of Information Processing in Medical Imaging*, pages 188–204, Springer-Verlag, Berlin, 1991.
- [8] M.J. Cambell and D. Machin. *Medical Statistics: A Commonsense Approach*. John Wiley and Sons, New York, NY, 1993.
- [9] I. Carlbom, D. Terzopoulos, and K.M. Harris. Computer-assisted registration, segmentation, and 3D reconstruction from images of neuronal tissue sections. *IEEE Transactions on Medical Imaging*, 13(2):351–362, Jun 1994.
- [10] G.I. Chiou and J.N. Hwang. A neural network-based stochastic active contour model (NNS-SNAKE) for contour finding of distinct features. *IEEE Transactions on Image Processing*, 4(10):1407–1416, Oct 1995.

-
- [11] H.E. Cline, C.L. Dumoulin, H.R. Hart Jr., W.E. Lorensen, and S. Ludke. 3D Reconstruction of the brain from magnetic resonance images using a connectivity algorithm. *Magnetic Resonance Imaging*, 5:245–352, Apr 1987.
- [12] H.E. Cline, W.E. Lorensen, R. Kikinis, and F. Jolesz. Three-dimensional segmentation of MR images of the head using probability and connectivity. *Journal of Computer Assisted Tomography*, 14(6):1037–1045, 1990.
- [13] A.R. Cohen, B. Roysam, and J.N. Turner. Automated tracing and volume measurements of neurons from 3D confocal fluorescence microscopy data. *Journal of Microscopy*, 173(2):103–114, Feb 1994.
- [14] I. Cohen, L.D. Cohen, and N. Ayache. Using deformable surfaces to segment 3-D images and infer differential structures. *Image Understanding*, 56(2):242–263, Sep 1992.
- [15] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1146, Nov 1993.
- [16] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, Jan 1995.
- [17] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, New York, NY, 1991.
- [18] C.A. Davatzikos and J.L. Prince. An active contour model for mapping the cortex. *IEEE Transactions on Medical Imaging*, 14(1):65–80, Mar 1995.
- [19] N. Deo and C. Pang. Shortest-path algorithms: Taxonomy and annotation. *Networks*, 14:275–323, 1984.
- [20] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [21] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, NY, 1973.
- [22] A.C.M. Dumay, M.N.A.J. Claessens, C. Roos, J.J. Gerbrands, and J.H.C. Reiber. Object delineation in noisy images by a modified policy-iteration method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):952–958, Sep 1992.
- [23] T.L. Faber, E.M. Stokely, R.M. Peshock, and J.R. Corbett. A model-based four-dimensional left ventricular surface detector. *IEEE Transactions on Medical Imaging*, 10(3):321–329, Sep 1991.

-
- [24] A.X. Falcão. Visualização de volumes aplicada à área médica. FEEC-UNICAMP, Feb 1993. Tese de Mestrado.
- [25] A.X. Falcão and J.K. Udupa. Segmentation of 3D objects using live-wire. In *Proceedings of SPIE on Medical Imaging*, volume 3034, Newport Beach, CA, February 1997.
- [26] A.X. Falcão, J.K. Udupa, S. Samarasekera, and B.E. Hirsch. User-steered image boundary segmentation. In *Proceedings of SPIE on Medical Imaging*, volume 2710, pages 278–288, Newport Beach, CA, Feb 1996.
- [27] 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*, 1996. A ser publicado.
- [28] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, NY, 1990.
- [29] G. Gerig, J. Martin, R. Kikinis, O. Kübler, M. Shenton, and F.A. Jolesz. Automating segmentation of dual-echo MR head data. In *Proceedings of Information Processing in Medical Imaging*, pages 175–187, Springer-Verlag, Berlin, 1991.
- [30] G.L. Gimel'farb and A.V. Zalesny. Low-level bayesian segmentation of piecewise-homogeneous noisy and textured images. *International Journal of Imaging Systems and Technology*, 3:227–243, 1991.
- [31] R.C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison Wesley, 1987.
- [32] A. Goshtasby and D.A. Turner. Segmentation of cardiac cine MR images for extraction of right and left ventricular chambers. *IEEE Transactions on Medical Imaging*, 14(1):56–64, Mar 1995.
- [33] D.C. Hemmy, F.W. Zonneveld, S. Lobregt, and K. Fukuta. A decade of clinical three-dimensional imaging: A review. Part I. historical development. *Investigative Radiology*, 29(4):489–496, 1994.
- [34] G. Herman. Discrete multidimensional jordan surfaces. *Graphical Models and Image Processing*, 54:507–515, 1992.
- [35] 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.
- [36] 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.

-
- [37] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.
- [38] A. Kaufmann. *Introduction to The Theory of Fuzzy Subsets, Vol. 1*. Academic Press, New York, NY, 1975.
- [39] Y. Kita and Y. Shirai. Extraction of accurate stomach contours from X-Ray images of barium-filled stomachs and its applications to detect potential abnormalities. *Graphical Models and Image Processing*, 53(5):447–456, Sep 1991.
- [40] O. Kübler, J. Ylä-Jääski, and E. Hildebrand. 3-D Segmentation and real time display of medical volume images. In *Proceedings of Computer Assisted Radiology*, pages 637–641, Springer-Verlag, Berlin, 1987.
- [41] K.F. Lai and R.T. Chin. Deformable contours: Modeling and Extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1084–1091, Nov 1995.
- [42] F. Leymarie and M.D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–633, Jun 1993.
- [43] S.P. Liou and R.C. Jain. An approach to three-dimensional image segmentation. *Computer Vision and Image Understanding*, 53(3):237–252, May 1991.
- [44] S. Lobregt and M. A. Viergever. A discrete dynamic contour model. *IEEE Transactions on Medical Imaging*, 14(1):12–24, 1995.
- [45] A. Low. *Introductory Computer Vision and Image Processing*. McGraw-Hill, New York, NY, 1991.
- [46] A. Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1(2):169–182, Aug 1972.
- [47] W. Menhardt. Iconic fuzzy sets for MR image segmentation. In *Proceedings of Medical Images: Formation, Handling, and Evaluation*, pages 579–591, Springer-Verlag, Berlin, 1992.
- [48] M.B. Merickel, T. Jackson, C. Carman, J.R. Brookeman, and C.R. Ayers. A multispectral pattern recognition system for the noninvasive evaluation of atherosclerosis utilizing MRI. In *Proceedings of 3D-Imaging in Medicine: Algorithms, Systems, and Applications*, pages 133–146, Springer-Verlag, Berlin, 1990.
- [49] D. Mintz. Robust consensus based edge detection. *Image Understanding*, 59(2):137–153, Mar 1994.

-
- [50] U. Montanari. On the optimal detection of curves in noisy pictures. *ACM Communications Society*, 14(5):335–345, 1971.
- [51] E.N. Mortensen and W.A. Barrett. Intelligent scissors for image composition. In *Proceedings of Computer Graphics (SIGGRAPH)*, pages 191–198, Los Angeles, CA, Aug 1995.
- [52] M. Moshfeghi, S. Ranganath, and K. Nawyn. Three-dimensional elastic matching of volumes. *IEEE Transactions on Image Processing*, 3(2):128–138, Mar 1994.
- [53] N.J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kauffman, Palo Alto, CA, 1980.
- [54] J. Nuyts, P. Suetens, A. Oosterlinck, M. De Roo, and L. Mortelmans. Delineation of ECT images using global constraints and dynamic programming. *IEEE Transactions on Medical Imaging*, 10(4):489–498, Dec 1991.
- [55] M. Ozkan, B.M. Dawant, and R.J. Maciunas. Neural-network-based segmentation on multi-modal medical images: A comparative and prospective study. *IEEE Transactions on Medical Imaging*, 12(3):534–544, Sep 1993.
- [56] K.P. Philip, E.L. Dove, D.D. McPherson, N.L. Gotteiner, M.J. Vonesh, W. Stanford, J.E. Reed, J.A. Rumberg, and K.B. Chandran. Automatic detection of myocardial contours in cine-computed tomographic images. *IEEE Transactions on Medical Imaging*, 13(2):241–253, Jun 1994.
- [57] S.M. Pizer, T.J. Cullip, and R.E. Fredericksen. Toward interactive object definition in 3D scalar images. In *Proceedings of 3D-Imaging in Medicine: Algorithms, Systems, and Applications*, pages 83–105, Springer-Verlag, Berlin, 1990.
- [58] D. Pope, D. Parker, D. Gustafson, and P. Clayton. Dynamic research algorithms in left ventricular border recognition and analysis of coronary arteries. In *IEEE Proceedings of Computers in Cardiology*, pages 71–75, 1984.
- [59] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1992.
- [60] S.V. Raman, S. Sarkar, and K.L. Boyer. Tissue boundary refinement in magnetic resonance images using contour-based scale space matching. *IEEE Transactions on Medical Imaging*, 10(2):109–121, Jun 1991.
- [61] S.P. Raya and J.K. Udupa. Low-level segmentation of 3-D magnetic resonance brain images - a rule-based system. *IEEE Transactions on Medical Imaging*, 9(3):327–337, 1990.

-
- [62] T.R. Reed and J.M.H. Du Buf. A review of recent texture segmentation and feature extraction techniques. *Image Understanding*, 57(3):359–372, May 1993.
- [63] A. Rosenfeld and A.C. Kak. *Digital Picture Processing, Vol. 2*. Academic Press, New York, NY, 1982.
- [64] R. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, New York, NY, 1992.
- [65] T. Schiemann, M. Bomans, U. Tiede, and K.H. Höhne. Interactive 3D-segmentation. In *Proceedings of Visualization in Biomedical Computing II*, pages 376–383, Chapel Hill, NC, 1992.
- [66] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing Analysis and Machine Vision*. Chapman & Hall Computing, New York, NY, 1993.
- [67] M. Sonka, M.D. Winniford, and S.M. Collins. Robust simultaneous detection of coronary borders in complex images. *IEEE Transactions on Medical Imaging*, 14(1):151–161, Mar 1995.
- [68] S.Raya. SOFTVU – A software package for multidimensional medical image analysis. In *Proceedings of SPIE*, volume 1232, pages 162–166, 1990.
- [69] L.H. Staib and J.S. Duncan. Boundary finding with parametrically deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1061–1075, Nov 1992.
- [70] H.S. Stiehl. 3-D Image understanding in radiology. *IEEE Engineering and Medical Biology Magazine*, 9(4):24–28, 1990.
- [71] G. Storvik. A bayesian approach to dynamic contours through stochastic sampling and simulated annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):976–986, Oct 1994.
- [72] M.R. Stytz, G. Frieder, and O. Frieder. Three-dimensional medical imaging: Algorithms and computer systems. *ACM Communications Society*, 23(4):421–499, Dec 1991.
- [73] B.D. Tachkray and A.C. Nelson. Semi-automatic segmentation of vascular network images using rotating structuring element (rose) with mathematical morphology and dual feature thresholding. *IEEE Transactions on Medical Imaging*, 12(3):385–392, Sep 1993.
- [74] S.L. Tanimoto. *The Elements of Artificial Intelligence*. Computer Science Press, Rockville, MD, 1987.

-
- [75] A. Taratorin and S. Sideman. Constrained detection of left ventricular boundaries from cine CT images of human hearts. *IEEE Transactions on Medical Imaging*, 12(3):521–533, Sep 1993.
- [76] J. K. Udupa. Interactive segmentation and boundary surface formation. *Computer Graphics and Image Processing*, 18:213–235, 1982.
- [77] J. K. Udupa. Multidimensional digital boundaries. *Graphical Models and Image Processing*, 56(4):311–323, 1994.
- [78] J. K. Udupa. Three-dimensional imaging techniques: A current perspective. *Academic Radiology*, 2:335–340, 1995.
- [79] J.K. Udupa. Computer aspects of 3D imaging in medicine: A tutorial. Technical Report MIPG 153, Medical Image Processing Group, Dept. of Radiology, Univ. of Pennsylvania, Philadelphia, PA, 1989.
- [80] J.K. Udupa and R. Gonçalves. Imaging transforms for visualizing surfaces and volumes. *Journal of Digital Imaging*, 6(4):213–236, Nov 1993.
- [81] J.K. Udupa, G. Herman, and eds. *3D Imaging in Medicine*. CRC Press, Boca Raton, FL, 1991.
- [82] J.K. Udupa, B.E. Hirsch, S. Samarasekera, and R.J. Gonçalves. Joint kinematics via three-dimensional MR imaging. In *SPIE Proceedings: Visualization in Biomedical Computing*, volume 1808, pages 664–669, Oct 1992.
- [83] J.K. Udupa, D. Odhner, S. Samarasekera, R.J. Gonçalves, K. Iyer, K. Venugopal, and S. Furuie. 3DVIEWNIX: An open, transportable, multidimensional, multimodality, multiparametric imaging software system. In *Proceedings of SPIE: Image Capture, Formatting, and Display*, volume 2164, pages 58–73, 1994.
- [84] J.K. Udupa and S. Samarasekera. Fuzzy connectedness and object definition. In *Proceedings of SPIE on Medical Imaging*, volume 2431, pages 2–11, San Diego, CA, Feb 1995.
- [85] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, Nov 1995.
- [86] M.W. Vannier, C.M. Speidel, D.L. Rickman, L.D. Schertz, L.R. Baker, C.F. Hildeboldt, C.J. Offutt, J.A. Balko, R.L. Butterfield, and M.H. Gado. Multispectral analysis of magnetic resonance images. In *IEEE Proceedings of 9th International Conference on Pattern Recognition*, pages 1182–1186, Washington, DC, 1988.

-
- [87] D. Williams and M. Shah. Edge characterization using normalized edge detector. *Graphical Models and Image Processing*, 55(4):311–318, Jul 1993.
- [88] D.J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *Image Understanding*, 55(1):14–26, Jan 1992.
- [89] S.R. Yhann and T.Y. Young. Boundary localization in texture segmentation. *IEEE Transactions on Image Processing*, 4(6):849–870, Jun 1995.
- [90] H.S. Zadeh, J.P. Windham, and F. Chen. Automated contour extracting using a multi-scale approach. *Computer Vision and Image Understanding*, 53(3):237–252, May 1991.
- [91] S. Zhan and R. Mehrotra. A zero-crossing-based optimal three-dimensional edge detector. *Image Understanding*, 59(2):242–253, Mar 1994.
- [92] F.W. Zonneveld. A decade of clinical three-dimensional imaging: A review. Part III. image analysis and interaction, display options, and physical models. *Investigative Radiology*, 29(7):716–725, 1994.
- [93] F.W. Zonneveld and K. Fukuta. A decade of clinical three-dimensional imaging: A review. Part II. clinical applications. *Investigative Radiology*, 29(5):574–589, 1994.