**Extensão da Transformada Imagem-Floresta
Diferencial para funções de conexidade com
aumentos baseados na raiz e sua aplicação
para geração de superpixels**

Marcos Ademir Tejada Condori

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Programa: Ciência da Computação

Orientador: Prof. Dr. Paulo André Vechiatto de Miranda

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES

São Paulo, dezembro de 2017

Extensão da Transformada Imagem-Floresta Diferencial para funções de conexidade com aumentos baseados na raiz e sua aplicação para geração de superpixels

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 11/12/2017. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Paulo André Vechiatto de Miranda (orientador) - IME-USP
- Prof. Dr. Fábio Augusto Menocci Cappabianco - UNIFESP
- Prof. Dr. Thiago Vallin Spina - UNICAMP

Agradecimentos

Agradeço a Deus pela oportunidade, saúde e força no decorrer do meu crescimento pessoal e profissional.

Aos meus pais Manuel e Irma e aos meus irmãos Alejandro e Maryori pelo amor, confiança e apoio que sempre me deram mesmo estando tão longe, mas sempre no meu coração. À minha namorada Giuliana por estar ao meu lado neste caminho.

Agradeço ao Prof. Dr. Paulo Miranda por compartilhar seus conhecimentos, pela compreensão e dedicação na realização desse trabalho.

Aos meus amigos e colegas do laboratório Lucy, Hans, Caio, Anderson, Mariela, Dênnis, Maciel e Raphael, amigos que estiveram ao meu lado durante esta fase, pela boa recepção, amizade e experiências.

Às minhas amigas do LIAMF, Viviane, Erika, Paula e Débora pelas conversas, convivências e amizades.

Agradeço à CAPES pelo apoio financeiro.

Resumo

Condori, M. A. T. **Extensão da Transformada Imagem-Floresta Diferencial para funções de conexidade com aumentos baseados na raiz e sua aplicação para geração de superpixels.** 2017. 104 f. Dissertação Mestrado - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

A segmentação de imagens é um problema muito importante em visão computacional, no qual uma imagem é dividida em regiões relevantes, tal como para isolar objetos de interesse de uma dada aplicação. Métodos de segmentação baseados na transformada imagem-floresta (IFT, *Image Foresting Transform*), com funções de conexidade *monotonicamente incrementais* (MI) têm alcançado um grande sucesso em vários contextos. Na segmentação interativa de imagens, na qual o usuário pode especificar o objeto desejado, novas sementes podem ser adicionadas e/ou removidas para corrigir a rotulação até conseguir a segmentação esperada. Este processo gera uma sequência de IFTs que podem ser calculadas de modo mais eficiente pela DIFT (Differential Image Foresting Transform). Recentemente, funções de conexidade *não monotonicamente incrementais* (NMI) têm sido usadas com sucesso no arcabouço da IFT no contexto de segmentação de imagens, permitindo incorporar informações de alto nível, tais como, restrições de forma, polaridade de borda e restrição de conexidade, a fim de customizar a segmentação para um dado objeto desejado. Funções não monotonicamente incrementais foram também exploradas com sucesso na geração de superpixels, via sequências de execuções da IFT.

Neste trabalho, apresentamos um estudo sobre a Transformada Imagem-Floresta Diferencial no caso de funções NMI. Nossos estudos indicam que o algoritmo da DIFT original apresenta uma série de inconsistências para funções não monotonicamente incrementais. Este trabalho estende a DIFT, visando incorporar um subconjunto das funções NMI em grafos dirigidos e mostrar sua aplicação no contexto da geração de superpixels. Outra aplicação que é apresentada para difundir a relevância das funções NMI é o algoritmo Bandeirantes para perseguição de bordas e rastreamento de curvas. **Palavras-chave:** transformada imagem-floresta, segmentação interativa, funções não monotonicamente incrementais, transformada imagem-floresta diferencial, rastreamento de curvas, perseguição de bordas, superpixels.

Abstract

Condori, M. A. T. **Extending the Differential Image Foresting Transform to connectivity functions with root-based increases and its application for superpixels generation.** 2017. 104 f. Dissertação Mestrado - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

Image segmentation is a problem of great relevance in computer vision, in which an image is divided into relevant regions, such as to isolate an object of interest for a given application. Segmentation methods with monotonically incremental connectivity functions (MI) based on the Image Foresting Transform (IFT) have achieved great success in several contexts. In interactive segmentation of images, in which the user is allowed to specify the desired object, new seeds can be added and/or removed to correct the labeling until achieving the expected segmentation. This process generates a sequence of IFTs that can be calculated more efficiently by the Differential Image Foresting Transform (DIFT). Recently, non-monotonically incremental connectivity functions (NMI) have been used successfully in the IFT framework in the context of image segmentation, allowing the incorporation of shape, boundary polarity, and connectivity constraints, in order to customize the segmentation for a given target object. Non-monotonically incremental functions were also successfully exploited in the generation of superpixels, via sequences of IFT executions.

In this work, we present a study of the Differential Image Foresting Transform in the case of NMI functions. Our research indicates that the original DIFT algorithm presents a series of inconsistencies for non-monotonically incremental functions. This work extends the DIFT algorithm to NMI functions in directed graphs, and shows its application in the context of the generation of superpixels. Another application that is presented to spread the relevance of NMI functions is the Bandeirantes algorithm for curve tracing and boundary tracking.

Keywords: image foresting transform, interactive segmentation, non-monotonically incremental functions, differential image foresting transform, curve tracing, boundary tracking, superpixels.

Sumário

Lista de Abreviaturas	xi
Lista de Símbolos	xiii
Lista de Figuras	xv
Lista de Tabelas	xvii
1 Introdução	1
1.1 Motivação e Objetivos	2
1.2 Contribuições	2
1.3 Organização do Trabalho	3
2 Conceitos	5
2.1 Imagem digital	5
2.2 Grafos	6
2.2.1 Caminho	6
2.3 Função de conexidade	7
2.3.1 Funções Monotonicamente Incrementais - MI	7
2.3.2 Caminho ótimo	8
2.4 Grafo derivado de imagens	8
2.5 Segmentação de imagens	8
2.5.1 Imagem de rótulos	9
2.6 Segmentação por competição de sementes	9
3 Transformada Imagem-Floresta	11
3.1 Definição	11
3.2 Algoritmo geral da IFT	12
3.3 Regiões de empate	16
3.3.1 Ordem de avaliação dos nós vizinhos	17
3.3.2 Políticas de desempate	17
4 Transformada Imagem-Floresta Diferencial	21
4.1 Definição	21
4.2 Equivalência de resultados	23
4.3 Regras de modificação da DIFT por Falcão e Bergo (2004)	23

4.3.1	Adição de novas sementes	23
4.3.2	Remoção de sementes	25
4.3.3	Combinação de operações, adição e remoção de sementes	25
4.4	Algoritmo da DIFT original [Ber04]	26
4.5	Teste de estados	27
4.5.1	Inconsistências com funções monotonicamente incrementais	29
5	Comportamento da DIFT* para funções de custo com aumentos baseados na raiz	31
5.1	Funções de conexidade com aumentos baseados na raiz	31
5.2	Inconsistências nos mapas de raízes e de rótulos	33
5.3	Inconsistências no mapa de custos	35
6	Algoritmo DIFTmod	37
6.1	Problema na inclusão do teste de estados	37
6.1.1	Proposta de solução para as inconsistências geradas pela inclusão do teste de estados, com funções MI	39
6.2	Solução para funções de custo com aumentos baseados na raiz	42
6.2.1	Algoritmo de poda	47
7	DIFTmod para a aplicação de geração de superpixels	51
7.1	Algoritmo IFT-SLIC	51
7.2	Modo diferencial da IFT-SLIC	53
7.3	DIFTmod para a geração de superpixels	56
7.4	Metodologia dos experimentos	56
7.4.1	Base de imagens	57
7.4.2	Parâmetros	57
7.4.3	Análise da acurácia	57
7.5	Resultados dos experimentos	59
7.5.1	Resultados da acurácia	59
7.5.2	Resultados do desempenho	60
8	Resultados experimentais para a função f_{euc}	63
8.1	Metodologia	63
8.2	Resultados de uma “série”	64
8.3	Resultados para diferentes topologias de grafos	67
9	Conclusões	71
9.1	Sugestões para Pesquisas Futuras	71
A	Bandeirantes	73
A.1	Motivação	73
A.2	Trabalhos relacionados	74
A.2.1	Funções de custo	75
A.3	Bandeirantes	76

A.4 Resultados experimentais e conclusões	77
Referências Bibliográficas	81

Lista de Abreviaturas

MI	Monotonicamente incrementais
NMI	Não monotonicamente incrementais
IFT	Tranformada imagem-floresta (<i>Image Foresting Transform</i>)
DIFT	Tranformada imagem-floresta diferencial (<i>Differential Image Foresting Transform</i>)

Lista de Símbolos

f	Função de conexidade
G	Grafo
V	Conjunto de vértices
E	Conjunto de arcos
C	Mapa de custos
P	Mapa de predecessores
L	Mapa de rótulos
R	Mapa de raízes
Q	Fila de prioridade
S	Conjunto de sementes
$\Delta_{S_i}^+$	Conjunto de sementes novas
$\Delta_{S_i}^-$	Conjunto de sementes removidas
$I(t)$	Intensidade do pixel t
π	Caminho
$\pi_{r \rightsquigarrow t}$	Caminho com origem em r até o spel t
Π	Conjunto de todos os caminhos
$\omega(\langle s, t \rangle)$	Valor associado ao arco entre os spels s e t
C_i	Cena resultante da i -ésima execução sequencial de IFTs

Lista de Figuras

1.1	Segmentação interativa por meio da DIFT	1
2.1	Imagem em tons de cinza	5
2.2	Imagem multibanda	6
2.3	Imagem multidimensional	6
2.4	Extensão de caminho	7
2.5	Grafo derivado de uma imagem 2D	8
2.6	Grafo derivado de uma imagem 3D	8
2.7	Segmentação interativa de imagens	9
2.8	Segmentação por competição de sementes	10
3.1	Mapas da IFT	12
3.2	Inicialização do algoritmo da IFT	14
3.3	Correspondência de cores adotadas	14
3.4	Execução do algoritmo da IFT	15
3.4	Execução do algoritmo da IFT, continuação	16
3.5	Regiões de empate	16
3.6	Ordem de avaliação dos nós adjacentes	17
3.7	Políticas de desempate FIFO e LIFO	18
3.8	Estrutura de dados Dial	19
4.1	Segmentação interativa	21
4.2	Representação de uma árvore com rótulo inconsistente e o teste do predecessor	24
4.3	Remoção de sementes e o conjunto de fronteira	25
4.4	Inconsistências nos mapas de rótulos e de raízes na DIFT*	30
5.1	Diagrama dos subconjuntos de interesse das funções de conexidade	32
5.2	Regiões desconexas no mapa de rótulos L	33
5.3	Exemplo de inconsistência no mapa de rótulos para a função f_{abs_add}	34
5.4	Exemplo de inconsistências geradas pelo algoritmo da DIFT*	35
5.5	Inconsistência no mapa de custos C	36
6.1	Inconsistência no mapa de rótulos na DIFT*	38
6.2	Modificação da política de inserção na fila de prioridades $\mathcal{Q}$	40
6.3	Representação do Algoritmo DIFTmod ^{FMI}	41
6.4	Inconsistência de rótulos para a função NMI f_{abs_add}	43

6.5	Solução de inconsistências para funções NMI	44
6.5	Solução de inconsistências para funções NMI, continuação	45
6.6	Comparação de resultados para a função f_{abs_add}	45
6.7	Execução da DIFTmod para a função f_{euc}	48
6.8	Resultados dos algoritmos para a função f_{euc}	49
7.1	Superpixels gerados pelo Algoritmo IFT-SLIC	52
7.2	Atualização de raízes IFT-SLIC	52
7.3	Processo diferencial de geração de superpixels	56
7.4	Base de imagens Grabcut	57
7.5	Correspondência de rótulos	58
7.6	Medida gráfica da acurácia dos algoritmos.	59
7.7	Percentagem de divergência entre os resultados dos algoritmos diferenciais (DIFT*, DIFTmod) e a IFTx10, para distintos valores dos parâmetros k e α	60
7.8	Desempenho dos Algoritmos DIFT*, DIFTmod e IFTx10 para distintos valores dos parâmetros k e α	61
8.1	Variação de sementes durante uma série.	64
8.2	Desempenho do algoritmo DIFTmod para a função f_{euc}	65
8.3	Divergência nos mapas R	65
A.1	Métodos baseados em grafos, Live-wire e Riverbed	75
A.2	Exemplo do rastreamento do contorno externo do cérebro	77
A.3	Perseguição de bordas do Bandeirantes	77
A.4	Exemplo da aplicação do Bandeirantes para avaliação automática de provas	78

Lista de Tabelas

4.1	Nós modificados pelos fluxos sequencial e diferencial	22
7.1	Regiões desconexas originadas pela DIFT* no Algoritmo IFT-SLIC com $k = 400$. . .	54
7.2	Regiões desconexas originadas pela DIFT*, no Algoritmo IFT-SLIC com $k = 800$. . .	55
7.3	Resultados sem inconsistências	60
8.1	Divergência dos mapas C e L entre a DIFT* e a IFT	66
8.2	Divergência dos mapas C e L entre a DIFTmod e a IFT	66
8.3	Divergência percentual dos mapas C e L na iteração 20.	67
8.4	Resultados de desempenho para a função f_{euc}	68
8.5	Resultados da divergência para a função f_{euc}	69
A.1	A taxa média de atribuições corretas das curvas por imagem para diferentes métodos.	79

Capítulo 1

Introdução

O processamento de imagens via Transformada Imagem-Floresta (Image Foresting Transform - IFT) [FSL04] possui várias aplicações bem sucedidas, incluindo filtragem [FCL01], perseguição de bordas [FUM00], segmentação [MM14] e classificação [PFMM07]. Esse algoritmo em grafos obtém uma floresta de caminhos ótimos, encontrando caminhos que minimizam os custos de uma função de custo monotonicamente incremental - MI. Na segmentação via IFT, os objetos são definidos pela floresta de caminhos ótimos cujas árvores estão enraizadas em suas sementes internas. Em segmentação interativa de imagens, novas sementes podem ser adicionadas e/ou removidas para corrigir erros na rotulação. Temos portanto, múltiplas execuções da IFT para cada novo conjunto de sementes. Ao invés de calcular cada IFT a partir do zero a cada interação, a Transformada Imagem-Floresta Diferencial (Differential Image Foresting Transform - DIFT) [FB04] permite que árvores sejam adicionadas ou removidas da floresta de uma maneira incremental (Figura 1.1). Com isso, temos um maior desempenho com execução em tempo sublinear, proporcional ao tamanho das regiões da floresta alteradas.

Figura 1.1: (a) Segmentação do cérebro em uma imagem de ressonância magnética. (b-e) Sementes são inseridas e removidas de modo a obter a segmentação do cerebelo.

As funções não monotonicamente incrementais - NMI compõem uma classe de funções de conectividade menos restrita, possibilitando configurações mais adaptativas, por exemplo, para lidar com problemas de inomogeneidade [Man14]. Historicamente, funções de conectividade NMI foram evitadas, dado que os caminhos gerados pela IFT com funções NMI, podem não ter otimalidade garantida de

acordo com a definição apresentada em [FSL04]. Em trabalhos anteriores a 2010, funções NMI foram consideradas apenas para calcular uma aproximação da distância Euclidiana [FCC02] [TFC04] e para reconstrução superior local [Fal09], sendo bem limitadas suas aplicações em segmentação de imagens, contando apenas com alguns trabalhos de caráter empírico [HC01] [MFaRB08].

Posteriormente, foi provado que algumas funções NMI levam a resultados ótimos de acordo com outros critérios de otimalidade, tais como uma função de energia para perseguição ótima de bordas [MFS11] [MFS12] ou uma medida de corte em grafos dirigidos para o particionamento ótimo do grafo [MM14][MM13b]. Com este embasamento teórico, a pesquisa envolvendo funções NMI atingiu um novo patamar.

O algoritmo original da DIFT apresenta uma série de inconsistências para funções não monotonicamente incrementais. Este trabalho estende a DIFT para um subconjunto das funções NMI (funções de conexidade com aumentos baseados na raiz) em grafos dirigidos.

Uma segunda contribuição desse trabalho corresponde a um novo método baseado em funções NMI, de uma subclasse de funções de conexidade com aumentos baseados no prefixo e não somente na raiz, chamado Bandeirantes [CMM17], para perseguição de bordas em segmentação e rastreamento de curvas, que foi comparado com Livewire [FUM00], Riverbed [MFS12] e G-Wire [Kan05] em uma aplicação para correção automática de provas.

1.1 Motivação e Objetivos

Recentemente, funções de conexidade NMI têm sido usadas com sucesso no arcabouço da IFT no contexto de segmentação de imagens [MMC13], permitindo incorporar restrições de alto nível, tais como, restrições de forma [MM13a], polaridade de borda [MM14] e restrições de conexidade [MMC16], a fim de customizar a segmentação para um dado objeto desejado.

Em outro trabalho, Strand et al. propuseram o método Minimum Barrier Distance (MBD) [SCMS13], o qual, em cenários digitais, leva a uma função NMI; que tem mostrado bons resultados em comparações empíricas.

Mais recentemente, a segmentação de múltiplos objetos satisfazendo restrições hierárquicas foi possível mediante o emprego de funções NMI [LM17]. Funções NMI foram também exploradas com sucesso na geração de superpixels [ACFM15], via sequências de execuções da IFT. No entanto, não existia um estudo sobre a Transformada Imagem-Floresta Diferencial para o cálculo eficiente de sequências de IFTs no caso de funções NMI. O presente trabalho procura preencher essa lacuna, estendendo a DIFT para um subconjunto de funções NMI (funções de conexidade com aumentos baseados na raiz) em grafos dirigidos, com resultados de aplicação na geração de superpixels.

1.2 Contribuições

As principais contribuições deste trabalho são as seguintes:

- Um primeiro estudo da DIFT para um subconjunto das funções NMI, envolvendo funções de conexidade com aumentos baseados na raiz.
- Desenvolvimento do algoritmo DIFTmod que integra um conjunto maior de funções para o cálculo diferencial eficiente no arcabouço da IFT, incluindo funções clássicas MI e funções NMI com aumentos baseados na raiz.
- Por meio do algoritmo proposto DIFTmod foi possível melhorar o desempenho de geração de superpixels do algoritmo IFT-SLIC, o qual é importante para uma ampla variedade de aplicações de superpixels.
- Desenvolvimento de uma aplicação com sucesso por meio de uma função de conexidade NMI com aumentos baseados no prefixo (conjunto de funções que até então não haviam sido exploradas na literatura) por meio do novo algoritmo Bandeirantes, para perseguição de bordas em segmentação e rastreamento de curvas [CMM17].

- Um dataset de imagens para rastreamento de curvas¹.
- Uma demonstração online do arcabouço da IFT incluindo a DIFTmod².
- Duas publicações em conferências, SIBGRAPI17 [CCFM17] e ISMM17 [CMM17].

1.3 Organização do Trabalho

No Capítulo 2 são apresentados os conceitos básicos e notações referentes a grafos e imagens a serem utilizadas no documento. Como base do trabalho, temos o método da IFT, detalhado no Capítulo 3 e os conceitos do processamento diferencial DIFT no Capítulo 4. No Capítulo 5 são mostradas as implicações de utilizar funções NMI (funções de conexidade com aumentos baseados na raiz) no algoritmo da DIFT.

As modificações necessárias para o desenvolvimento do novo algoritmo DIFTmod são apresentadas no Capítulo 6. A aplicação da DIFTmod na geração de superpixels, bem como os experimentos e resultados são expostos no Capítulo 7. Experimentos da DIFTmod com a função f_{euc} no contexto da Transformada de Distância Euclidiana são apresentados no Capítulo 8. Finalmente no Capítulo 9 discutimos as conclusões e sugestões de trabalhos futuros. No Apêndice A é mostrado o novo método para perseguição de curvas chamado Bandeirantes [CMM17].

¹<http://www.vision.ime.usp.br/~mtejadac/bandeirantes.html>

²<http://www.vision.ime.usp.br/~mtejadac/demos.html>

Capítulo 2

Conceitos

Neste capítulo são apresentados os conceitos utilizados ao longo do trabalho assim como a notação utilizada.

2.1 Imagem digital

Uma imagem digital $\hat{I}$ é um par $(\mathcal{I}, \vec{I})$ no qual $\mathcal{I}$ é um conjunto de pontos do Z^n (domínio da imagem) denominados spels¹ e $\vec{I}(a)$ é um mapeamento vetorial que associa a cada spel $a \in \mathcal{I}$ um vetor de valores escalares $\vec{I}(a) = \{I_1(a), I_2(a), \dots, I_m(a)\} \in Z^m$, associados a um conjunto de m propriedades físicas. O valor de n refere-se à dimensão da imagem e o valor de m ao número de bandas. Segundo os parâmetros de uma imagem digital, temos:

Imagem em tons de cinza:

Uma imagem em tons de cinza (Figura 2.1) ou de escala monocromática $\hat{I} = (\mathcal{I}, I)$, tem domínio bidimensional ($\mathcal{I} \in Z^2$) com apenas uma banda ($m = 1$), por exemplo, imagens de ultrassom e fatias tomográficas.

Uma imagem em tons de cinza é portanto representada como uma matriz de tamanho $M \times N$ (M linhas e N colunas)².

Figura 2.1: Representação de uma imagem em tons de cinza

Imagem multibanda:

Uma imagem digital $\hat{I} = (\mathcal{I}, \vec{I})$ é multibanda ou multiespectral se $\hat{I}$ associa a cada spel a , mais de um valor escalar, ($\vec{I}(a) = \{I_1(a), I_2(a), \dots, I_m(a)\}$, para $m > 1$). Por exemplo, as imagens de satélite conseguem obter valores de intensidade em distintos comprimentos de onda (visíveis e não visíveis), uma imagem colorida (Figura 2.2) capturada por uma câmera comum possui um $m = 3$ correspondendo aos componentes vermelho, verde e azul.

¹spels (*spatial elements*), no caso particular de imagens 2D e 3D são chamados *pixel* e *voxel* respectivamente.

²Uma imagem de dimensões 640×480 será representada por uma matriz de ordem 480×640 .

Figura 2.2: Imagem multibanda, cada banda corresponde a uma componente (vermelho, verde e azul).

Imagem multidimensional:

Uma imagem multidimensional $\hat{I} = (\mathcal{I}, \vec{I})$ define seu domínio de amostragem $I \subset Z^n$, para $n > 2$. Por exemplo, as imagens multidimensionais que fazem um volume no espaço (imagens de tomografias) (Figura 2.3), ou também um arquivo de vídeo que é composto por uma sequência de imagens (quadros do vídeo).

Figura 2.3: Imagem multidimensional em tons de cinza.

2.2 Grafos

Um grafo G é um par $\langle V, E \rangle$, no qual V representa um conjunto finito não vazio de elementos denominados *vértices* ou *nós* e $E \subseteq V \times V$ é um conjunto de pares de elementos $\langle u, v \rangle$ pertencentes a V denominados *arcos*.

Um grafo G é chamado dirigido (digrafo ou orientado) quando o conjunto E está composto de pares *ordenados*, os arcos $\langle u, v \rangle$ e $\langle v, u \rangle$ são considerados distintos, caso contrário o grafo é chamado de não dirigido ou não orientado. No presente trabalho usamos apenas o conceito de grafo dirigido.

Subgrafo Um grafo $G' = \langle V', E' \rangle$ é chamado subgrafo de $G = \langle V, E \rangle$ se $V' \subseteq V$ e $E' \subseteq E$.

Adjacência Dizemos que um vértice v é adjacente a um vértice u , se existe um arco que sai do nó u e entra em v . Nessas circunstâncias, dizemos também que v é um vizinho de u . Usamos $\mathcal{A}(u) = \{v \in V | \langle u, v \rangle \in E\}$ para indicar o conjunto de vértices adjacentes a u .

2.2.1 Caminho

Um caminho $\pi = \langle t_1, \dots, t_n \rangle$ é definido como uma sequência de vértices distintos e adjacentes (isto é, $\langle t_i, t_{i+1} \rangle \in E, i = 1, 2, \dots, n-1$). Usamos a notação $\pi_t = \langle t_1, \dots, t_n = t \rangle$ para indicar um caminho com término no vértice t e $\pi_{r \rightsquigarrow t} = \langle t_1 = r, \dots, t_n = t \rangle$ para indicar um caminho com origem em r e término em t .

Um caminho composto por um único vértice é denotado por $\pi_t = \langle t \rangle$, também denominado *caminho trivial*. A notação $\pi_s \cdot \langle s, t \rangle$ é usada para indicar a extensão do caminho π_s pelo arco $\langle s, t \rangle \in E$ (Figura 2.4). Empregamos a notação Π para denotar o conjunto de todos os possíveis caminhos no grafo.

Figura 2.4: Extensão do caminho π_s pelo arco $\langle s, t \rangle$.

2.3 Função de conexidade

Uma função de conexidade $f : \Pi \rightarrow \mathcal{C}$, também chamada *função de custo*, associa a qualquer caminho π_t no grafo, um valor $f(\pi_t)$ pertencente a um conjunto $\mathcal{C}$ totalmente ordenado de valores. Normalmente f é baseada nas propriedades locais e globais dos nós e arcos que compõem o caminho.

2.3.1 Funções Monotonicamente Incrementais - MI

Uma função é chamada Monotonicamente Incremental quando ela pode ser representada da seguinte forma:

$$\begin{aligned} f(\langle s \rangle) &= H(s) \\ f(\pi_s \cdot \langle s, t \rangle) &= f(\pi_s) \odot \langle s, t \rangle \end{aligned} \quad (2.1)$$

na qual $H(s)$ é um valor de custo inicial e $\odot : \mathcal{C} \times \mathbf{E} \rightarrow \mathcal{C}$ é uma operação binária que satisfaz as seguintes condições:

$$\text{M1. } f(\pi') \geq f(\pi) \Rightarrow f(\pi') \odot \langle s, t \rangle \geq f(\pi) \odot \langle s, t \rangle$$

$$\text{M2. } f(\pi) \odot \langle s, t \rangle \geq f(\pi)$$

para qualquer caminho $\pi \in \Pi$ e $\langle s, t \rangle \in \mathbf{E}$.

Exemplos de funções MI são apresentados nas Equações 2.2 e 2.3.

$$H(s) = \begin{cases} 0 & , \text{ se } s \in \mathcal{S} \\ +\infty & , \text{ caso contrário} \end{cases}$$

$$\begin{aligned} f_{sum}(\langle s \rangle) &= H(s) \\ f_{sum}(\pi_s \cdot \langle s, t \rangle) &= f_{sum}(\pi_s) + \omega(\langle s, t \rangle) \end{aligned} \quad (2.2)$$

$$\begin{aligned} f_{max}(\langle s \rangle) &= H(s) \\ f_{max}(\pi_s \cdot \langle s, t \rangle) &= \max\{f_{max}(\pi_s), \omega(\langle s, t \rangle)\} \end{aligned} \quad (2.3)$$

nas quais $\omega(\langle s, t \rangle)$ representa um peso não negativo associado aos arcos do grafo (note que $\omega(\langle s, t \rangle)$ pode ou não ser igual a $\omega(\langle t, s \rangle)$) e $\mathcal{S}$ representa um conjunto de vértices sementes, usados para restringir a origem dos caminhos.

2.3.2 Caminho ótimo

Um caminho π_t é *ótimo* se $f(\pi_t) \leq f(\tau_t)$, para qualquer outro caminho τ_t , independentemente de sua origem. Para cada nó $t \in V$, temos um valor único $C^*(t)$ que armazena o valor de um caminho ótimo com término em t , sendo C^* o mapa de custos ótimos.

$$C^*(t) = \min_{\pi_t \in \Pi} \{f(\pi_t)\}$$

2.4 Grafo derivado de imagens

Uma imagem pode ser interpretada como um grafo $G = \langle V, E \rangle$, no qual os vértices são os spels da imagem $\mathcal{I}$ e os arcos são determinados por uma *relação de adjacência* $\mathcal{A}$.

As posições de uma imagem $\mathcal{I}$, definem pontos num espaço Z^n , em que n refere-se à dimensão da imagem. Por exemplo, podemos usar a relação de adjacência Euclidiana para definir os arcos, baseados em uma constante ρ e na posição relativa de cada vértice, como mostrado na Figura 2.5. Um arco $\langle s, t \rangle$ é criado se a distância Euclidiana entre os nós s e t for menor ou igual à constante ρ .

Figura 2.5: Obtenção de um grafo a partir de uma imagem 2D por meio da função de distância Euclidiana e uma constante ρ . (a) $\rho = 1$, vizinhança 4. (b) $\rho = \sqrt{2}$, vizinhança 8. (c) $\rho = 2$, vizinhança 12.

Similarmente o conceito é extensível para imagens multidimensionais (3D), na Figura 2.6 são mostrados os vizinhos do vértice vermelho para distintos valores de ρ .

Figura 2.6: Geração de um grafo a partir de uma imagem multidimensional em Z^3 . (a) $\rho = 1$, vizinhança 6. (b) $\rho = 1.5$, vizinhança 18. (c) $\rho = \sqrt{3}$, vizinhança 26.

2.5 Segmentação de imagens

A segmentação de imagens é o processo de dividir ou classificar os pixels de $\mathcal{I}$ em duas ou mais classes, atribuindo rótulos para cada pixel da imagem, tal que pixels com o mesmo rótulo idealmente correspondem a um mesmo objeto presente na imagem.

A *segmentação interativa* de imagens, permite que o usuário especifique o objeto desejado das formas a seguir:

- Rotulando alguns pixels (chamados de sementes) dentro e fora do objeto, como por exemplo no método da Transformada Imagem-Floresta Orientada (OIFT) [MMC16] (Figura 2.7a).
- Indicando os limites do objeto na imagem, como por exemplo no método GrabCut [RKB04] (Figura 2.7b).
- Marcando pontos no contorno do objeto desejado, como por exemplo no método Live-Wire [FUS+98] (Figura 2.7c).

Neste trabalho, consideramos o primeiro enfoque para segmentar uma imagem. Visto que métodos baseados em sementes são mais facilmente extensíveis para imagens multidimensionais, além de serem menos sensíveis a pequenas variações da posição quando comparados a seleção de pontos no contorno.

(a) Sementes no objeto e no fundo, segmentação binária

(b) Caixa delimitadora

(c) Pontos no contorno do objeto

Figura 2.7: Seleção de objetos na segmentação de imagens.

2.5.1 Imagem de rótulos

Uma imagem de rótulos L é definida como uma função $L : \mathcal{I} \rightarrow \{0, \dots, l - 1\}$, na qual l é o número de classes/clusters na imagem. Em uma segmentação baseada em sementes/marcadores, o conjunto de sementes é representado por $\mathcal{S}$ e λ é a função de rotulação parcial inicial, que define o rótulo inicial das sementes $\lambda : \mathcal{S} \rightarrow \{0, \dots, l - 1\}$. Essa rotulação parcial da imagem é então propagada para os demais pixels, seguindo algum critério para partição ótima da imagem.

2.6 Segmentação por competição de sementes

Na segmentação de imagens por meio da IFT (Transformada Imagem Floresta) é possível realizar segmentações para mais de duas classes (imagens com múltiplos objetos). Consideramos assim conjuntos iniciais de vértices chamados de *sementes*. Por exemplo, no caso de uma segmentação binária $\lambda(s) \in \{0, 1\}$ para qualquer $s \in \mathcal{S}$, temos os conjuntos $\mathcal{S}_o$ composto por todas as sementes rotuladas como objeto (*object*) e o conjunto $\mathcal{S}_b$ composto por todas as sementes rotuladas como fundo (*background*), tal que $\mathcal{S}_o \cap \mathcal{S}_b = \emptyset$ e $\mathcal{S}_o \cup \mathcal{S}_b = \mathcal{S}$ (Figura 2.8). Neste paradigma os marcadores iniciais competem pela conquista dos pixels na imagem, oferecendo um custo segundo a função de conexidade de modo que o grafo é particionado em florestas enraizadas nas sementes.

Figura 2.8: Representação de uma segmentação binária objeto/fundo por meio da IFT. (a) Imagem com marcadores amarelos e roxos representando o objeto e o fundo respectivamente. (b) Resultado da segmentação e as florestas enraizadas nas sementes.

Capítulo 3

Transformada Imagem-Floresta

A Transformada Imagem Floresta (*Image Foresting Transform* - IFT) [FSL04] é um arcabouço de desenvolvimento de operadores baseados em conexidade para processamento de imagens, incluindo filtros morfológicos [LFZ02], rastreamento de curvas e perseguição de bordas [FUM00] [CMM17], segmentação por região [MM14] e classificação de padrões [CFYU12]. Neste capítulo apresentamos o algoritmo da IFT e seus conceitos principais.

3.1 Definição

A IFT é uma extensão do algoritmo de Dijkstra [Dij59] para várias fontes e funções de custo mais gerais [FSL04]. Ela reduz problemas de processamento de imagens baseados em conexidade ao cálculo de uma floresta de caminhos ótimos no grafo derivado da imagem, seguido de um pós-processamento simples de atributos na floresta resultante.

Uma imagem pode ser interpretada como um grafo dirigido $G = \langle \mathcal{I}, \mathbf{E} \rangle$, no qual os nós são os pixels da imagem, no domínio da imagem $\mathcal{I} \subset Z^2$, e os arcos são pares ordenados de pixels $\langle s, t \rangle \in \mathbf{E}$ (Seção 2.4). Os arcos podem ter valores de peso $\omega(\langle s, t \rangle)$ que dependem das propriedades locais da imagem, tais como *cor*, *gradiente*, *posição do pixel*, representando assim um *grafo ponderado*.

O algoritmo da IFT será apresentado com base nas seguintes definições:

Mapa de predecessores Um mapa de predecessores é uma função $P : \mathcal{I} \rightarrow \mathcal{I} \cup \{\text{nil}\}$ que atribui para cada nó t em $\mathcal{I}$, algum outro nó adjacente $s \in \mathcal{A}(t)$, ou uma marca distintiva $\text{nil} \notin \mathcal{I}$, caso em que t é dito ser uma raiz do mapa. Uma floresta de espalhamento corresponde a um mapa de predecessores que não contém ciclos (Figura 3.1(a)). Para qualquer pixel $t \in \mathcal{I}$, uma floresta de espalhamento P define um caminho π_t^P recursivamente como $\langle t \rangle$ se $P(t)$ é nil , e $\pi_s^P \cdot \langle s, t \rangle$ se $P(t) = s \neq \text{nil}$ (Figura 3.1(b)).

Mapa de raízes O mapa de raízes $R : \mathcal{I} \rightarrow \mathcal{I}$, relaciona cada spel $s \in \mathcal{I}$ a origem do caminho π_s^P na floresta gerada pela IFT. Isto é $\pi_s^P = \langle R(s), \dots, P(P(s)), P(s), s \rangle$ e $P(R(s)) = \text{nil}$. Neste trabalho, o mapa de raízes assume a forma $R : \mathcal{I} \rightarrow \mathcal{S}$, dado que restringimos os caminhos calculados para caminhos com origem no conjunto fornecido de pixels sementes $\mathcal{S}$ (Figura 3.1(c)).

Mapa de custo O mapa de custos $C : \mathcal{I} \rightarrow \mathcal{C}$, é uma função que associa cada spel $s \in \mathcal{I}$ ao custo do caminho π_s^P , na floresta gerada pela IFT (isto é, $C(s) = f(\pi_s^P)$).

Mapa de rótulos O mapa de rótulos $L : \mathcal{I} \rightarrow \{0, \dots, l-1\}$, é uma função que associa cada spel $s \in \mathcal{I}$ ao rótulo da sua raiz $r = R(s)$ (isto é, $\lambda(r) = L(r) = L(s)$) (Figura 3.1(d)).

Figura 3.1: (a) Mapa de predecessores P . (b) Ilustração do caminho π_t^P representado no mapa de predecessores, em que $P(t) = s$ e $P(r_0) = nil$. (c) Mapa de raízes R . (d) Mapa de rótulos do grafo, depois de uma segmentação binária.

3.2 Algoritmo geral da IFT

O algoritmo da IFT é apresentado a seguir (Algoritmo 1). Assumindo, sem perda de generalidade, que os custos finais calculados são todos finitos, custos iniciais infinitos não são inseridos na fila de prioridades a fim de otimizar o código. Como resultado do algoritmo temos o mapa de rótulos L representando uma segmentação de imagem, o mapa de custos C , o mapa de raízes R e o mapa de predecessores P contendo a floresta resultante.

Algoritmo 1: Algoritmo geral da IFT

Entrada : Imagem $\hat{I} = (\mathcal{I}, \vec{I})$, relação de adjacência $\mathcal{A}$, função de custo monotonicamente incremental f e uma função de rotulação inicial λ .

Saída : Floresta P de caminhos ótimos, mapa C de custos mínimos, mapa L de rótulos e o mapa R de raízes.

Auxiliares: Fila de prioridade $\mathcal{Q}$, variável tmp e um mapa de estado dos vértices.

```

1 para cada  $s \in \mathcal{I}$  faça
2    $P(s) \leftarrow nil, L(s) \leftarrow \lambda(s), R(s) \leftarrow s$ 
3    $C(s) \leftarrow f(\langle s \rangle)$ 
4    $estado(s) \leftarrow branco$ 
5   se  $C(s) \neq +\infty$  então Insira  $s$  em  $\mathcal{Q}$ 
6 enquanto  $\mathcal{Q} \neq \emptyset$  faça
7   Remova  $s$  de  $\mathcal{Q}$  cujo valor  $C(s)$  seja mínimo
8    $estado(s) \leftarrow preto$ 
9   para cada  $t \in \mathcal{A}(s)$  faça
10    se  $estado(t) \neq preto$  então
11      Calcule  $tmp \leftarrow f(\pi_s^P \cdot \langle s, t \rangle)$ 
12      se  $tmp < C(t)$  então
13        se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
14         $P(t) \leftarrow s, L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
15         $C(t) \leftarrow tmp$  e insira  $t$  em  $\mathcal{Q}$ 

```

Nas Linhas 2-4 são inicializados os mapas P, L, R, C e o mapa de *estado* dos nós. Na Linha 5 são inseridas as sementes na fila de prioridade $\mathcal{Q}$, dessa forma temos florestas de árvores triviais (enraizadas nas sementes) e nós isolados a serem conquistados durante o algoritmo.

Em cada iteração do laço principal (Linhas 6-15) é selecionado um nó s de custo mínimo em $\mathcal{Q}$. Este custo corresponde ao caminho ótimo até o nó s . Como o custo do nó s já contém seu melhor valor, o *estado* de s é alterado. Na Linha 9 são calculadas as possíveis extensões do caminho π_s^P pela regra de expansão da função de conexidade $f(\pi_s^P \cdot \langle s, t \rangle)$. Se algum custo calculado for melhor que o atual para algum nó $t \in \mathcal{A}(s)$ então são atualizados os mapas do nó t e sua posição na fila de prioridade com o novo custo $C(t)$. O Algoritmo 1 retorna essencialmente o mapa de predecessores P que representa a floresta ótima enraizada nas sementes e o mapa de rótulos L que simboliza a segmentação gerada pela função de custo f .

O teste de estados “ $estado(t) \neq preto$ ” (Linha 10), não está presente no algoritmo original da IFT, isto é porque para funções MI o teste de estados é considerado apenas para fins de melhoria do desempenho (evita a computação do custo do caminho da Linha 11) [FSL04]. Para funções de conexidade *não monotonicamente incrementais* (NMI) o teste de estados é necessário para garantir que cada nó seja processado apenas uma vez, de modo a evitar ciclos infinitos durante a execução do algoritmo [MM14].

A seguir, as Figuras 3.2 e 3.4, mostram uma execução do algoritmo da IFT com uma relação de adjacência de vizinhança 4 e a função de conexidade f_{max} (Equação 2.3). A Figura 3.2(a) representa uma imagem em tons de cinza e o seu grafo derivado é calculado segundo a regra $\omega(\langle s, t \rangle) = \lfloor \frac{I(s)-I(t)}{2} \rfloor$ (Figura 3.2(b)). Na Figura 3.2(c) temos a inicialização dos mapas P, L e C , representando cada um dos nós como árvores triviais. As sementes/raízes são representadas na forma de losango. À direita da Figura 3.2(c) é mostrada a fila de prioridade $\mathcal{Q}$ com as três raízes

iniciais, inseridas com um custo segundo a função f_{max} (Equação 2.3).

Figura 3.2: (a) Imagem em tons de cinza. (b) Grafo derivado da imagem com vizinhança 4 e peso nos arcos segundo a regra $\omega(\langle s, t \rangle) = \lfloor \frac{I(s) - I(t)}{2} \rfloor$. (c) Inicialização dos mapas P , L e C . As sementes são representadas com forma de losango e a direita temos a fila de prioridades Q .

Nas Figuras 3.4(a-p) são mostradas as 16 iterações do algoritmo da IFT. Note que o número de iterações corresponde ao número total de nós do grafo. Por exemplo, na Figura 3.4(a) o nó s com índice 2 é processado, e para cada um dos seus vizinhos em $\mathcal{A}$ é computado o valor de custo resultante da extensão do caminho da f_{max} (setas na cor vermelho). Como o valor oferecido é menor que o valor atual para cada vizinho (nós com índices 1, 3, 6) todos eles são conquistados por s e são inseridos na fila de prioridades Q . O próximo nó a ser processado será o nó com índice 12 (nó em cinza no topo da fila de prioridades). Na seguinte iteração do laço principal (Figura 3.4(b)) o nó 12 é avaliado, note que os nós previamente conquistados (nós 1, 3 e 6) estão na fila de prioridades e têm como predecessor o nó de índice 2.

Cada nó da Figura 3.4 tem uma cor correspondente a seu estado na fila de prioridades Q . Nós brancos representam nós que não foram inseridos na fila, nós amarelos estão na fila de prioridades e os nós pretos são os nós que já saíram da fila. Além disso o mapa de *estado* também é representado neste esquema de cores, sendo que nós com estado “branco” correspondem a nós que ainda não foram avaliados e nós com estado “preto” corresponde a nós que já foram avaliados pelo laço principal (Linhas 6-15) do Algoritmo 1 (Figura 3.3).

Figura 3.3: Correspondência de cores, adotadas na Figura 3.4 entre a fila de prioridades Q e o mapa de estados.

A cada iteração, cada rótulo (cores do fundo de cada nó) é propagado do predecessor ao novo nó conquistado. Ao final do algoritmo (Figuras 3.4(p-q)), obtemos três árvores enraizadas em cada semente. O resultado pode ser interpretado como uma segmentação da imagem, na qual nós com o mesmo rótulo pertencem à mesma classe/objeto.

Figura 3.4: parte I Execução do algoritmo da IFT (primeiros nós avaliados). (a) O nó s é selecionado para ser avaliado e muda seu estado para “preto”. O nó s computa o custo das extensões de caminho para seus vizinhos e os conquista. (b) O nó com índice 12 é avaliado, note que os nós 1, 3 e 6 previamente conquistados estão na fila de prioridades. (c) O nó com índice 15 é avaliado, note que a ordem na fila de prioridades foi alterada pela atualização anterior (Figura (b)). Nas figuras (d-p) são mostradas as iterações posteriores.

Note que o algoritmo pode ser interpretado como uma conquista de nós na qual é feita uma propagação de rótulos. Na Figura 3.4(h) o nó com índice 8 (rótulo verde) oferece um custo menor (custo 3) ao nó com índice 4 (o nó com índice 4 tem rótulo laranja, custo 11, seu predecessor é nó com índice 0 e sua raiz é o nó com índice 3), posto que o novo custo é melhor, o nó com índice 4 é

Figura 3.4: *continuação (q) Resultado da segmentação com as três sementes.*

conquistado. A Figura 3.4(i) mostra a atualização dos mapas para o nó com índice 4, rótulo verde, custo 3 (atualização na fila de prioridades), seu predecessor agora é o nó com índice 8 e sua raiz é o nó com índice 12).

3.3 Regiões de empate

No algoritmo da IFT, a floresta resultante pode não ser única. No caso de funções MI, empates entre caminhos provenientes de sementes com mesmo rótulo não representam um problema do ponto de vista da segmentação resultante. Problemas surgem quando temos duas sementes com rótulos distintos ($r_1 \neq r_2$ e $\lambda(r_1) \neq \lambda(r_2)$) e dois caminhos ótimos ($\pi_{r_1 \rightsquigarrow s}$ e $\pi_{r_2 \rightsquigarrow s}$) que oferecem o mesmo custo ao nó s (Figura 3.5(a)). Qualquer extensão de caminho com término em s ($\pi_s \cdot \langle s, t \rangle$) pode ter como raiz qualquer uma das sementes r_1 ou r_2 (Figura 3.5(b)(c)), o que representa um problema de ambiguidade. Esta região é chamada de *região de empate* [FSL04] (tie-zone).

Figura 3.5: (a) Mapa parcial de rótulos L no qual as sementes r_1 e r_2 conquistaram grande parte do grafo até o nó s . (b-c) Possíveis resultados que dependem da raiz que conquistou em primeiro lugar o nó s . A região disputada é uma região de empate.

Para tratar estas ambiguidades, são definidos os seguintes comportamentos:

1. A ordem de avaliação dos nós vizinhos t , na adjacência $\mathcal{A}(s)$ (Linha 9 do Algoritmo 1).
2. Uma política de desempate da fila de prioridades $\mathcal{Q}$.

3.3.1 Ordem de avaliação dos nós vizinhos

Na Linha 9 da IFT (Algoritmo 1) os nós t vizinhos do nó s são visitados, tal que $\langle s, t \rangle \in \mathcal{A}$. Mas a ordem em que eles podem ser processados pode ser variada (Figura 3.6), produzindo diferentes florestas. No presente trabalho a ordem de visita dos nós adjacentes é no sentido *raster*, mostrado na Figura 3.6(a).

Figura 3.6: Ordem de avaliação dos nós adjacentes (a) Avaliação em sentido raster. (b) Avaliação em sentido horário. (c) Avaliação em sentido anti-horário.

3.3.2 Políticas de desempate

Outra ambiguidade é resolvida definindo a política de desempate na fila de prioridades $\mathcal{Q}$. A fila de prioridades segue uma política de desempate *First In First Out* (FIFO) quando é escolhido o primeiro nó de menor custo inserido em $\mathcal{Q}$; ou segue uma política de desempate *Last Input First Output* (LIFO) quando é escolhido o último nó de menor custo que foi inserido na fila. No entanto, para conservar a consistência de escolher sempre o último caminho nas zonas de empates, para a política LIFO, devemos mudar o sinal de estritamente menor da Linha 12 do Algoritmo 1 para menor ou igual ($\leq$) [Man14].

Um exemplo das diferentes políticas de desempate, pode ser visto na Figura 3.7, para diferentes relações de adjacência.

Figura 3.7: Políticas de desempate. Representação da segmentação de um grafo com todos seus arcos com valor 4, a função de conexidade f_{max} (Equação 2.3) e as duas políticas de desempate.

Considerando os resultados da Figura 3.7, podemos afirmar que uma política de desempate FIFO produz uma segmentação mais equitativa que a política LIFO para regiões que não contém informação relevante nos arcos (regiões sem contraste na imagem original).

Estrutura da fila de prioridades

A fila de prioridades é uma das partes mais importantes do algoritmo da IFT. Uma implementação eficiente depende do tipo de dado resultante da função de conexidade. Por exemplo, se o resultado de $f(\pi_s^P) \in \mathbb{R}^+$, então $\mathcal{Q}$ pode ser implementada usando um heap binário. Para funções nas quais o custo de $f(\pi_s^P) \in \mathbb{I}^+$ e estão no intervalo $[0, \dots, K]$, foi proposto uma fila de Dial [Dia69]. A estrutura é um vetor circular de $K + 1$ buckets (Figura 3.8), no qual nós p com custo $C(p)$ são inseridos na posição $C(p) \% (K + 1)$. Cada bucket contém uma lista duplamente ligada dos nós com mesmo custo em C . Por exemplo, na Figura 3.8(a) os caminhos π_{p_2} e π_{p_5} tem um custo de caminho igual a 1. Além disso, cada bucket armazena dois ponteiros, um para o nó inserido com maior anti-guidade (Figura 3.8(a), seta azul), e outro ponteiro para o último nó que foi inserido (Figura 3.8(a), seta vermelha). Estes ponteiros são muito úteis para implementar as políticas de desempate. Nas Figuras 3.8(b-c) são representadas as políticas de desempate nas quais o seguinte nó a ser extraído é destacado em verde. As setas pretas horizontais indicam o sentido da inserção (na lista duplamente ligada para o bucket i) e setas verticais a extração (nó com o mínimo valor em $\mathcal{Q}$) dos nós da fila.

Figura 3.8: Estrutura de dados Dial para custos inteiros no mapa de custo C . (a) Representação da fila circular, com listas duplamente ligadas de nós com o mesmo custo. Em (b) e (c) é mostrado o próximo nó (destacado em verde) a ser avaliado para as políticas de desempate FIFO e LIFO respectivamente.

Capítulo 4

Transformada Imagem-Floresta Diferencial

No contexto de algum processo que implique em uma sequência de execuções de IFTs, com modificações no conjunto de sementes em cada iteração (feitas por um usuário ou alguma tarefa automatizada), é possível reduzir o tempo de execução da cadeia de IFTs, atualizando unicamente um subconjunto de caminhos a cada iteração. No cenário de segmentação interativa, por exemplo, sequências de IFTs são utilizadas a medida que o usuário marca sementes novas para corrigir a segmentação (Figura 4.1). Neste capítulo são apresentados os conceitos da Transformada Imagem-Floresta Diferencial (*Differential Image Florest Transform - DIFT*) [FB04] que atualiza um subconjunto dos caminhos com base no resultado anterior, restrito às regiões que tiveram seus rótulos e/ou custos modificados gerando um grande ganho do tempo de execução, o que é fundamental na segmentação de grandes volumes 3D.

4.1 Definição

Seja uma sequência de execuções de IFTs representada como $\langle IFT_{(\mathcal{S}_1)}, IFT_{(\mathcal{S}_2)}, \dots, IFT_{(\mathcal{S}_n)} \rangle$, na qual n é o número total de execuções sobre uma imagem $\hat{I}$. A cada execução, o conjunto de sementes $\mathcal{S}_i$ é modificado por meio de uma adição, remoção, ou uma combinação de adição e remoção de sementes para obter um novo conjunto $\mathcal{S}_{i+1}$. Definimos uma cena $\mathcal{C}_i$ como o conjunto de mapas $\mathcal{C}_i = \{P_i, C_i, L_i, R_i\}$, resultantes da i -ésima iteração em uma sequência de execuções de IFTs.

Figura 4.1: Segmentação interativa com o usuário.

A DIFT permite calcular eficientemente a cena C_i aproveitando os mapas da cena anterior C_{i-1} . Este cálculo é baseado num conjunto de regras que dependem das modificações feitas no conjunto de sementes [Ber04], alterando apenas um subconjunto de nós do grafo. Por exemplo, na Tabela 4.1 são mostrados os conjuntos de nós que foram modificados em três iterações de uma segmentação interativa. Na linha central da Tabela 4.1 (Figuras d-f) é apresentada a aplicação sequencial de múltiplas IFTs independentes, que em cada uma das iterações calcula todos os caminhos desde o princípio sobre a imagem toda, desconsiderando assim toda informação calculada em iterações anteriores. Na linha inferior da Tabela 4.1 (Figuras g-i) é apresentado o fluxo diferencial por meio do algoritmo da DIFT. Observe que apenas uma pequena porção dos pixels da imagem são reprocessados no fluxo diferencial.

	S_1	S_2	S_3
Resultado	(a)	(b)	(c)
Fluxo de IFTs independentes	(d)	(e)	(f)
Fluxo diferencial	(g)	(h)	(i)

Tabela 4.1: As regiões coloridas representam nós que foram modificados em cada execução para os fluxos diferencial e sequencial. Os marcadores destacados em vermelho em (a-c) representam as modificações para cada conjunto de sementes.

4.2 Equivalência de resultados

Seja $\mathcal{S}_i$ o conjunto de sementes da i -ésima iteração. O conjunto das novas sementes inseridas na próxima iteração é dado por $\Delta_{\mathcal{S}_{i+1}}^+ = \mathcal{S}_{i+1} \setminus \mathcal{S}_i$ e o conjunto das sementes removidas é $\Delta_{\mathcal{S}_{i+1}}^- = \mathcal{S}_i \setminus \mathcal{S}_{i+1}$.

Temos portanto, a seguinte sequência de IFTs no caso de n execuções independentes:

$$\langle IFT_{(\mathcal{S}_1)}, IFT_{(\mathcal{S}_2)}, \dots, IFT_{(\mathcal{S}_n)} \rangle$$

No fluxo diferencial da DIFT, após a primeira execução da $IFT_{(\mathcal{S}_1)}$, temos que as cenas $\mathcal{C}_i$ para $i \geq 2$ são calculadas com base na cena $\mathcal{C}_{i-1}$, aproveitando as árvores que foram calculadas na iteração anterior, reduzindo assim o tempo de processamento. Logo temos o seguinte fluxo diferencial:

$$\langle IFT_{(\mathcal{S}_1)}, DIFT_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, \mathcal{C}_1)}, DIFT_{(\Delta_{\mathcal{S}_3}^+, \Delta_{\mathcal{S}_3}^-, \mathcal{C}_2)}, \dots, DIFT_{(\Delta_{\mathcal{S}_n}^+, \Delta_{\mathcal{S}_n}^-, \mathcal{C}_{n-1})} \rangle$$

O fluxo diferencial deve obter um resultado válido, possível de ser obtido pela $IFT_{(\mathcal{S}_n)}$. Ou seja, a $IFT_{(\mathcal{S}_n)}$ pode ter vários resultados possíveis dependendo da ordem de processamento dos nós e arcos no caso de empates (Seção 3.3), mas o resultado do fluxo diferencial deve corresponder a um deles, logo diferenças na rotulação final devem ser restritas apenas dentro das regiões de empate (tie-zones).

Usaremos o símbolo de equivalência para denotar um resultado válido, note também que de modo mais geral temos:

$$\langle IFT_{(\mathcal{S}_1)}, \dots, IFT_{(\mathcal{S}_n)} \rangle = IFT_{(\mathcal{S}_n)} \equiv \langle IFT_{(\mathcal{S}_1)}, DIFT_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, \mathcal{C}_1)}, \dots, DIFT_{(\Delta_{\mathcal{S}_n}^+, \Delta_{\mathcal{S}_n}^-, \mathcal{C}_{n-1})} \rangle$$

Como consequência temos que, quando o resultado da $IFT_{(\mathcal{S}_n)}$ possui como saída um mapa de custo C_n único (ótimo ou não ótimo) o fluxo diferencial deve reproduzir exatamente esse mesmo mapa de custo. Dado que para funções MI, a IFT gera o mapa de custo ótimo que é único, o fluxo diferencial para funções MI deve gerar o mesmo mapa de custo ótimo.

4.3 Regras de modificação da DIFT por Falcão e Bergo (2004)

A seguir são apresentadas as regras de modificação para o fluxo diferencial com $n \geq 2$, assumindo que se tem uma primeira cena $\mathcal{C}_1$, previamente calculada. Discutimos abaixo apenas o caso de funções MI, pois a DIFT original [Ber04], foi projetada exclusivamente para essas funções.

4.3.1 Adição de novas sementes

A inserção de novas sementes em uma floresta de caminhos ótimos existente, pode ser feita inserindo as novas sementes $r' \in \Delta_{\mathcal{S}_{i+1}}^+$ na fila de prioridade $\mathcal{Q}$ (atualmente vazia) e executando o laço principal do Algoritmo 1 da IFT, sem o teste de estado da Linha 10.

Suponha a inserção de uma nova semente r' durante uma iteração (Figura 4.2(b)), tal que, $\pi_{r' \rightsquigarrow s}$ torna-se um caminho ótimo para o nó s (incluindo o caso $s = r'$). Se o novo caminho, para s é estendido pelo arco $\langle s, t \rangle$, oferecendo ao nó t um custo menor, $f(\pi_{r' \rightsquigarrow s} \cdot \langle s, t \rangle) < C_i(t)$, então, o caminho π_t^P será atualizado como $\pi_{r' \rightsquigarrow s} \cdot \langle s, t \rangle$ (Figura 4.2(c)) e t será inserido na fila de prioridades $\mathcal{Q}$. Porém, no caso em que o custo oferecido de um caminho com origem em r' é igual ao custo atual de algum nó, por exemplo nó u , $f(\pi_{r' \rightsquigarrow t} \cdot \langle t, u \rangle) = C_i(u)$ (Figura 4.2(d)), o nó u não será conquistado pelo caminho $\pi_{r' \rightsquigarrow t}$, segundo o algoritmo da IFT. Este comportamento pode gerar uma *árvore com rótulos inconsistentes* (regiões com uma variação de rótulos dentro de uma mesma árvore, ver Figura 4.2(e)). Para corrigir esse problema, Falcão e Bergo [FB04] [Ber04] propuseram a inclusão do *teste do predecessor* (Linha 14 do Algoritmo 2).

O teste do predecessor evita a formação de árvores com rótulos inconsistentes, mediante a condição $P(u) = t$ (Linha 14 do Algoritmo 2). Considerando o exemplo anterior, se o custo oferecido pelo novo caminho $\pi_{r' \rightsquigarrow t}$ for igual ao custo atual do nó u , então se o teste do predecessor, $P(u) = t$, for verdadeiro o nó u será conquistado pelo caminho $\pi_{r' \rightsquigarrow t}$. Ao final da avaliação do nó t , é garantido que se $P(u) = t$, então $L(u) = L(t)$ para todos os vizinhos u , tal que $\langle t, u \rangle \in \mathcal{A}$ (Figura 4.2(f)).

Figura 4.2: Teste do predecessor. (a) Imagem segmentada com duas sementes. (b) Inserção da nova semente r' . (c) A semente r' conquista os nós para os quais oferece um custo menor (Linha 12 do Algoritmo 1), como por exemplo, o nó t . (d) O caminho $\pi_{r' \rightsquigarrow t}$ oferece um custo igual ao custo atual do nó u , então u não é modificado. (e) A subárvore com raiz em u mantém seu rótulo anterior (um rótulo errado), formando dessa maneira uma árvore com rótulo inconsistente. (f) Teste do predecessor, a partir da imagem “d”. O nó t oferece um custo igual ao custo atual do nó u , mas este é conquistado pelo teste do predecessor, evitando a formação da árvore com rótulo inconsistente.

O teste do predecessor procura conservar a coerência dos caminhos novos em relação aos mapas P_{i-1} e L_{i-1} da iteração anterior. Os caminhos novos devem manter uma propagação correta de rótulos com relação a caminhos existentes. Na Figura 4.2 é mostrado o comportamento do teste do predecessor, no qual, apesar de que o custo oferecido pelo nó t ao nó u não ser melhor que o seu custo atual, o nó u é conquistado pelo caminho enraizado em r' .

4.3.2 Remoção de sementes

Uma das propriedades para modificar/corrigir uma segmentação de modo interativo, é ter a capacidade de remover sementes (no grafo, remover árvores) de regiões que tenham sido rotuladas em uma iteração anterior. Para o fluxo sequencial, a remoção de alguma semente implica em uma nova execução da IFT sobre a imagem toda. Com o fluxo diferencial é possível remover árvores da floresta existente e calcular novos caminhos de custo ótimo de maneira incremental.

Seja, o nó $r_3 \in \Delta_{S_{i+1}}^-$ selecionado para remoção, a árvore enraizada na semente r_3 tem que ser removida (Figura 4.3(b)). Para conquistar as regiões removidas, o algoritmo da DIFT cria um conjunto de nós de fronteira $\mathcal{F}$ (região amarela na Figura 4.3(c)), composto por nós que tem algum vizinho enraizado nas sementes que foram removidas, logo os nós de fronteira são usados como pontos de partida para uma nova propagação de rótulos (setas na Figura 4.3(c)). Ao final as regiões removidas devem ser reconquistadas por algum caminho ótimo enraizado nas sementes não removidas (Figura 4.3(d)).

Figura 4.3: Representação do conjunto de fronteira $\mathcal{F}$. (a) Resultado final de uma segmentação com três sementes. (b) A região r_3 é selecionada para ser removida. (c) A árvore enraizada em r_3 é removida e é criado o conjunto de fronteira $\mathcal{F}$ (região amarela). (d) Resultado da propagação de caminhos ótimos a partir dos nós do conjunto de fronteira.

Os nós u que não foram removidos continuam presentes com seus valores antigos nos mapas $C(u)$, $L(u)$, $P(u)$ e $R(u)$. Os nós de fronteira são encontrados no Algoritmo 3 DIFT-LimpaÁrvores, realizando uma busca em largura, a partir das raízes das árvores removidas, andando na contramão no mapa de predecessores P e visitando os nós adjacentes. Neste laço se algum nó visitado não pertence a uma região removida, este é adicionado ao conjunto de nós de fronteira $\mathcal{F}$ [Ber04].

4.3.3 Combinação de operações, adição e remoção de sementes

Na combinação de operações de adição de sementes ($\Delta_{S_{i+1}}^+ \neq \emptyset$) e remoção de regiões ($\Delta_{S_{i+1}}^- \neq \emptyset$) em uma mesma execução, algumas considerações devem ser tomadas:

- Se alguma semente $s \in \Delta_{S_{i+1}}^+$ coincide com um nó de fronteira, depois da criação de $\mathcal{F}$, devemos removê-lo de $\mathcal{F}$ e inicializar seus atributos como atributos próprios de uma semente nos mapas L , C , R e P . Note que se seus atributos não fossem atualizados, o comportamento de s seria de um caminho estendido, propagando informação desatualizada e incorreta.
- O conjunto de fronteira $\mathcal{F}$, deve ser criado antes que as novas sementes em $\Delta_{S_{i+1}}^+$ sejam inicializadas, pois esta inicialização modifica os mapas de rótulos L e o mapa de predecessores

P , o que pode tornar alguns nós inatingíveis para o algoritmo de criação do conjunto $\mathcal{F}$ (Algoritmo 3).

4.4 Algoritmo da DIFT original [Ber04]

Algoritmo 2: Algoritmo DIFT original [Ber04]

Entrada : Imagem $\hat{I}$, relação de adjacência $\mathcal{A}$, o conjunto de sementes a adicionar Δ_S^+ , conjunto de sementes a remover Δ_S^- , função de custo monotonicamente incremental f , os mapas P , L , R , C e a função de rotulação λ

Saída : Mapas P , L , R , C .

Auxiliares: Fila de prioridades $\mathcal{Q}$, conjunto de nós de fronteira $\mathcal{F}$

```

1  $\mathcal{Q} \leftarrow \emptyset$ 
2  $(P, L, R, C, \mathcal{F}) \leftarrow \text{DIFT-LimpaÁrvores}(P, L, R, C, \mathcal{A}, \Delta_S^-)$ 
3  $\mathcal{F} \leftarrow \mathcal{F} \setminus \Delta_S^+$ 
4 para cada  $s \in \mathcal{F}$  faça
5    $\mid$  Insira  $s$  em  $\mathcal{Q}$  com custo  $C(s)$ 
6 para cada  $s \in \Delta_S^+$  faça
7    $\mid$  se  $f(\langle s \rangle) < C(s)$  então
8      $\mid \mid P(s) \leftarrow \text{nil}, L(s) \leftarrow \lambda(s), R(s) \leftarrow s$ 
9      $\mid \mid C(s) \leftarrow f(\langle s \rangle)$  e insira  $s$  em  $\mathcal{Q}$  com custo  $C(s)$ 
10 enquanto  $\mathcal{Q} \neq \emptyset$  faça
11    $\mid$  Remova  $s$  de  $\mathcal{Q}$  tal que  $C(s)$  seja mínimo
12    $\mid$  para cada  $t \in \mathcal{A}(s)$  faça
13      $\mid \mid$  Calcule  $\text{tmp} \leftarrow f(\pi_s^P \cdot \langle s, t \rangle)$ 
14      $\mid \mid$  se  $\text{tmp} < C(t)$  ou  $s = P(t)$  então
15        $\mid \mid \mid$  se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
16        $\mid \mid \mid P(t) \leftarrow s, L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
17        $\mid \mid \mid C(t) \leftarrow \text{tmp}$  e insira  $t$  em  $\mathcal{Q}$ 

```

No Algoritmo 2 da DIFT, a Linha 2 remove as árvores com a origem em Δ_S^- e constrói o conjunto de fronteira $\mathcal{F}$ por meio do Algoritmo 3 (DIFT-LimpaÁrvores). Além disso, retorna os mapas P , L , R e C atualizados. A Linha 3, remove as novas sementes do conjunto de fronteira (Subseção 4.3.3). Na Linha 5, são inseridos os nós de fronteira na fila de prioridades $\mathcal{Q}$ com o custo atual de cada um deles. Nas Linhas 6-9 são inicializadas as novas sementes e são inseridas em $\mathcal{Q}$ (atualmente vazia). O laço das Linhas 10-17 é semelhante ao laço principal (sem o **teste de estados**) da IFT, para cada nó s que sai da fila são avaliados seus vizinhos $t \in \mathcal{A}(s)$, se o custo oferecido pela extensão de caminho $f(\pi_s^P \cdot \langle s, t \rangle)$ é menor ao custo atual de t ou se o **teste do predecessor** é verdadeiro então os mapas de t são atualizados e t é inserido na fila de prioridades.

Algoritmo 3: Algoritmo DIFT-LimpaÁrvores [Ber04]

Entrada : Relação de adjacência $\mathcal{A}$, conjunto de sementes a serem removidas Δ_S^- e os mapas P , L , R e C .

Saída : Mapas P , L , R , C e o conjunto de fronteira $\mathcal{F}$.

Auxiliares: Fila FIFO T .

```

1  $\mathcal{F} \leftarrow \emptyset$ 
2 para cada  $r \in \Delta_S^-$  faça
3    $P(r) \leftarrow nil, L(r) \leftarrow nil, R(r) \leftarrow nil$ 
4    $C(r) \leftarrow +\infty$  e insira  $r$  em  $T$ 
5 enquanto  $T$  não estiver vazia faça
6   Remova  $s$  de  $T$ 
7   para cada  $t \in \mathcal{A}(s)$  faça
8     se  $P(t) = s$  então
9        $P(t) \leftarrow nil, C(t) \leftarrow +\infty$ 
10      Insira  $t$  em  $T$ 
11     senão
12       se  $R(t) \notin \Delta_S^-$  então
13          $\mathcal{F} \leftarrow \mathcal{F} \cup \{t\}$ 

```

Na Linha 1 do Algoritmo 3 DIFT-LimpaÁrvores, é inicializado o conjunto $\mathcal{F}$. Nas Linhas 2-4 as raízes no conjunto Δ_S^- , são inicializadas como árvores triviais com custo infinito e são inseridas na fila T . O laço das Linhas 5-13 propaga a remoção da árvore num percurso em largura (desde as raízes). Dita propagação, continua enquanto um vizinho t é sucessor do nó s ($P(t) = s$), o que implica que eles pertencem à mesma árvore. Caso $P(t) = s$, o nó t é inserido na fila T , para continuar a propagação (Linha 10). Ao longo da remoção das árvores, os nós que pertencem às regiões removidas são convertidos em árvores triviais (Linha 9), para que possam ser conquistados pelas novas sementes em Δ_S^+ ou pelos nós de fronteira em $\mathcal{F}$. Se o nó t não pertence a alguma árvore a ser removida, então, t é inserido no conjunto de fronteira $\mathcal{F}$ (Linha 13).

4.5 Teste de estados

O “teste de estados” discutido na Seção 3.2, atualmente não faz parte do Algoritmo 2 da DIFT, tal como proposto em [Ber04]. Porém seria interessante incorporá-lo, pois o “teste de estado” é fundamental no caso de funções NMI. A seguir apresentamos o Algoritmo DIFT* (que é o Algoritmo DIFT, modificado para introduzir o teste de estados) e mostramos que somente a introdução do “teste de estado” não é suficiente para garantir a corretude do algoritmo diferencial no caso de funções NMI (Capítulo 5). Na verdade, a sua introdução gera problemas inclusive em casos envolvendo funções MI (Seção 4.5.1).

Algoritmo 4: Algoritmo DIFT*

Entrada : Imagem $\hat{I}$, relação de adjacência $\mathcal{A}$, o conjunto de sementes a adicionar Δ_S^+ , conjunto de sementes a remover Δ_S^- , função de conexidade f , os mapas P , L , R , C e a função de rotulação λ .

Saída : Mapas P , L , R e C .

Auxiliares: Fila de prioridades $\mathcal{Q}$ e o conjunto de nós de fronteira $\mathcal{F}$

```

1  $\mathcal{Q} \leftarrow \emptyset$ 
2 para cada  $p \in \mathcal{I}$  faça  $estado(p) \leftarrow branco$ 
3 se  $\Delta_S^- \neq \emptyset$  então
4    $(P, L, R, C, \mathcal{F}) \leftarrow \text{DIFT-LimpaÁrvores}(P, L, R, C, \mathcal{A}, \Delta_S^-)$ 
5    $\mathcal{F} \leftarrow \mathcal{F} \setminus \Delta_S^+$ 
6   para cada  $s \in \mathcal{F}$  faça
7      $\text{Insira } s \text{ em } \mathcal{Q} \text{ com custo } C(s)$ 
8 para cada  $s \in \Delta_S^+$  faça
9   se  $f(\langle s \rangle) < C(s)$  então
10      $P(s) \leftarrow nil, L(s) \leftarrow \lambda(s), R(s) \leftarrow s$ 
11      $C(s) \leftarrow f(\langle s \rangle)$  e insira  $s$  em  $\mathcal{Q}$  com custo  $C(s)$ 
12 enquanto  $\mathcal{Q} \neq \emptyset$  faça
13   Remova  $s$  de  $\mathcal{Q}$  tal que  $C(s)$  seja mínimo
14    $estado(s) \leftarrow preto$ 
15   para cada  $t \in \mathcal{A}(s)$  faça
16     se  $estado(t) \neq preto$  então
17       Calcule  $tmp \leftarrow f(\pi_s^P \cdot \langle s, t \rangle)$ 
18       se  $tmp < C(t)$  ou  $s = P(t)$  então
19         se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
20          $P(t) \leftarrow s, L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
21          $C(t) \leftarrow tmp$  e insira  $t$  em  $\mathcal{Q}$  com custo  $C(s)$ 

```

No Algoritmo 4 da DIFT*, todos os nós são inicializados em um estado “*branco*”, o que indica que não foram avaliados nem pertencem à fila de prioridades $\mathcal{Q}$ (Linha 2). Após gerar o conjunto de fronteira (Linha 4). Os nós em $\mathcal{F}$ são inseridos na fila de prioridades (Linhas 6-7), assim como as novas sementes (Linha 11) após serem inicializadas (Linha 10).

Uma das modificações principais está na Linha 14, na qual o estado do nó s que saiu da fila de prioridades $\mathcal{Q}$ é transformado em um estado “*preto*”, o que indica que o nó já contém seu caminho definitivo, e que portanto não pode ser alterado na mesma iteração. Na Linha 16 é avaliado o **teste de estados**, o qual garante que sejam avaliados unicamente nós com um estado distinto do “*preto*”.

Como foi mencionado na Seção 3.3, a IFT pode produzir múltiplas soluções, no caso de empates. Apesar disso para funções MI, os algoritmos DIFT e DIFT* têm que obter o mesmo mapa de custo. Mas, a inclusão do “teste de estados”, Algoritmo 4, pode ocasionar inconsistências no mapa de raízes R e no mapa de rótulos L .

Inconsistência no mapa de raízes R : Consideramos um mapa de raízes inconsistente, se existe algum nó $s \in \mathcal{I}$ tal que $\pi_s^P = \langle t_1, \dots, t_n = s \rangle$ e $R(s) \neq t_1$.

Inconsistência no mapa de rótulos L : Consideramos um mapa de rótulos L inconsistente, se existe algum nó $s \in \mathcal{I}$ tal que $\pi_s^P = \langle t_1, \dots, t_n = s \rangle$ e $L(t_1) \neq L(s)$.

4.5.1 Inconsistências com funções monotonicamente incrementais

Com a inclusão do “teste de estados” e sob certas circunstâncias, é possível gerar uma inconsistência no mapa de rótulos considerando o Algoritmo 4 da DIFT*. Na Figura 4.4 é apresentado este comportamento, utilizando a vizinhança 4, uma política de desempate FIFO e a função de conexidade f_{max} (Equação 2.3).

Dado o grafo G (Figura 4.4(a)), foi executada uma primeira $IFT_{(\mathcal{S}_1)}$ com duas sementes. Como resultado foi obtida uma cena C_1 (Figura 4.4(b)), na qual as sementes estão representadas com a forma de losangos. Em uma segunda iteração ($DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$), a árvore de rótulo laranja é selecionada para ser removida ao mesmo tempo que uma nova semente é inserida no canto superior esquerdo (Figura 4.4(c)). Durante a execução da DIFT*, o Algoritmo 3 (DIFT-LimpaÁrvores), remove a região laranja e cria o conjunto de fronteira $\mathcal{F}$. Os nós do conjunto de fronteira e a nova semente são inseridos na fila de prioridades (Figura 4.4(d)). Nós de cor amarelo são usados para indicar quais nós estão na fila de prioridades $\mathcal{Q}$. Na Figura 4.4(e) os nós s e t têm o mesmo custo. Dado que $\mathcal{Q}$ tem uma política FIFO o nó s é avaliado antes que o nó t (porque s foi inserido com antecedência como parte de $\mathcal{F}$), além disso, no exemplo o nó s não consegue oferecer um custo melhor a nenhum de seus vizinhos e muda para o estado “preto”.

Posteriormente quando o nó t é avaliado (Figura 4.4(f)), embora t seja o predecessor do nó s , o caminho π_t^P não pode conquistar o nó s (porque s tem um estado “preto”). Neste instante é possível observar que os nós t e s apesar de terem a mesma raiz, não têm o mesmo rótulo. Como resultado da execução da $DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$ (Figura 4.4(g)) temos uma subárvore com raiz no nó s com o rótulo distinto do rótulo da sua raiz (inconsistências no mapa de raízes). A Figura 4.4(h) mostra o resultado obtido pela $IFT_{(\mathcal{S}_2)}$. Como pode-se observar nos resultados dos algoritmos, os mapas de custo são iguais mas os mapas de rótulos e de raízes da $DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$ têm inconsistências. A proposta de solução para estes problemas (funções MI) é apresentado na Seção 6.1.1.

Figura 4.4: Algoritmo da $DIFT^*$, inconsistências nos mapas de rótulos e de raízes para a função f_{max} (Equação 2.3), política de desempate FIFO e uma relação de adjacência de vizinhança 4. (a) Grafo derivado de uma imagem. (b) Resultado da $IFT_{(S_1)}$ com duas sementes com os nós semente em forma de losango. (c) Remoção da árvore de rótulo laranja e inserção de uma nova semente (rótulo azul) no canto superior esquerdo (d) O conjunto $\mathcal{F}$ é criado e seus nós são inseridos na fila de prioridades $\mathcal{Q}$, acompanhados da nova semente. Os nós com a cor amarela representam vértices que estão na fila de prioridades. (e) O nó s é avaliado e muda seu estado para “preto”. (f) O nó t não pode conquistar o nó s (mesmo sendo seu predecessor), s e t não tem o mesmo rótulo embora $R(s) = R(t)$. (g) Propagação das inconsistências. (h) Resultado da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$. Note as inconsistências de rótulos para a subárvore com a origem no nó s e as inconsistências no mapa de raízes. (i) Resultado obtido por meio da $IFT_{(S_2)}$.

Capítulo 5

Comportamento da DIFT* para funções de custo com aumentos baseados na raiz

O algoritmo original da DIFT foi desenvolvido e tem sido aplicado com sucesso para funções monotonicamente incrementais - MI. No entanto, para funções NMI para os quais o aumento de custo do caminho depende de alguma propriedade da raiz, é possível obter resultados inconsistentes, nos mapas de rótulos L , de raízes R e no mapa de custos C . O objetivo deste trabalho é estender o algoritmo da DIFT, para manter os resultados atuais e suportar esta nova classe de funções. Neste capítulo são apresentadas as inconsistências que foram estudadas para distintas funções de custo com aumentos baseados na raiz.

5.1 Funções de conexidade com aumentos baseados na raiz

Definição 1. Função de conexidade com aumentos baseados na raiz. Definimos uma função de conexidade com aumentos baseados na raiz, como uma função na qual na extensão de um caminho $\pi_{r \rightsquigarrow s}$ pelo arco $\langle s, t \rangle$, tem-se que $\Delta_C = f(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) - f(\pi_{r \rightsquigarrow s})$ é uma função não negativa g que depende apenas de r , s e t . Isto é $\Delta_C = g(r, s, t) \geq 0$.

Note que funções de custo com aumentos baseados na raiz são um caso particular de uma classe mais geral de funções com aumentos baseados no prefixo (Figura 5.1).

Definição 2. Função de conexidade com aumentos baseados no prefixo. Definimos uma função de custo com aumentos baseados no prefixo, como uma função na qual na extensão de um caminho $\pi_{t_{n-1}} = \langle t_1, \dots, t_{n-1} \rangle$ pelo arco $\langle t_{n-1}, t_n \rangle$, tem-se que $\Delta_C = f(\pi_{t_{n-1}} \cdot \langle t_{n-1}, t_n \rangle) - f(\pi_{t_{n-1}})$ é uma função não negativa que depende de todos nós $t_1, \dots, t_n$. Isto é $\Delta_C = g(t_1, \dots, t_n) \geq 0$.

Figura 5.1: Diagrama dos subconjuntos de interesse das funções de conectividade.

Nesta seção estamos interessados em funções de conectividade com aumentos baseados na raiz. No Apêndice A será mostrado um exemplo prático de uma função com aumentos baseados no prefixo que não depende apenas da raiz. Como exemplos de funções de conectividade com aumentos baseados na raiz temos a função f_D (Equação 5.1) utilizada para a geração de Superpixels [ACFM15] e a função f_{euc} (Equação 5.2) para o cálculo da Transformada de Distância Euclidiana - EDT.

$$\begin{aligned} f_D(\pi_s = \langle s \rangle) &= \begin{cases} 0 & , \text{ se } s \in \mathcal{S} \\ +\infty & , \text{ caso contrário} \end{cases} \\ f_D(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) &= f_D(\pi_{r \rightsquigarrow s}) + (\|I(r) - I(t)\| \cdot \alpha)^\beta + \|t - s\| \end{aligned} \quad (5.1)$$

$$\begin{aligned} f_{euc}(\pi_t = \langle t \rangle) &= \begin{cases} 0 & \text{if } t \in \mathcal{S} \\ +\infty & \text{caso contrário} \end{cases} \\ f_{euc}(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) &= \|t - r\|^2 \end{aligned} \quad (5.2)$$

$$\begin{aligned} f_{abs_add}(\pi_t = \langle t \rangle) &= \begin{cases} 0 & \text{if } t \in \mathcal{S} \\ +\infty & \text{caso contrário} \end{cases} \\ f_{abs_add}(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) &= f_{abs_add}(\pi_{r \rightsquigarrow s}) + \|I(r) - I(t)\| + 1 \end{aligned} \quad (5.3)$$

Nas equações apresentadas, a extensão do caminho depende de uma propriedade da raiz (intensidade do pixel ou da posição relativa da raiz na imagem). Cabe mencionar que a Equação 5.3 é uma simplificação da Equação 5.1, que será utilizada em vários exemplos.

Note que na função f_{euc} , embora seu $\Delta_C = f_{euc}(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) - f_{euc}(\pi_{r \rightsquigarrow s}) = \|t - r\|^2 - \|s - r\|^2$ dependa da posição da raiz, ele pode ser negativo. O Δ_C pode ser negativo se o pixel t estiver mais próximo da raiz do que s . No entanto, na prática para os grafos derivados de imagens usados, os custos processados pela IFT com a função f_{euc} em geral possuem $\Delta_C \geq 0$, pois arcos $\langle s, t \rangle$ com t mais próximo da raiz em geral não acontecem, pois t já teria sido conquistado por outro caminho mais curto, estando portanto no estado **preto**.

Como nossa finalidade é calcular uma sequência de IFTs, de um modo diferencial, nas seguintes seções são apresentadas as inconsistências que são geradas pelo algoritmo da DIFT* para funções de custo com aumentos baseados na raiz.

5.2 Inconsistências nos mapas de raízes e de rótulos

As inconsistências nos mapas de raízes R e de rótulos L estão muito relacionadas entre si. A Figura 5.2 mostra um exemplo no qual as regiões desconexas têm um rótulo distinto da sua raiz no mapa L segundo o mapa de predecessores.

Figura 5.2: Imagem com inconsistências de raiz e de rótulo, as regiões indicadas pelas setas têm um rótulo distinto da sua raiz no mapa de rótulos segundo o mapa de predecessores.

Na Figura 5.3, é mostrado como é originada uma inconsistência de rótulo para a função f_{abs_add} (Equação 5.3).

Seja o grafo G derivado de uma imagem $\hat{I}$, no qual o valor em cada nó representa a intensidade do pixel (Figura 5.3(a)). Foram selecionadas duas sementes iniciais. O resultado obtido da execução da $IFT_{(S_1)}$ é uma floresta de caminhos enraizados nas duas sementes (Figura 5.3(b)), sendo as sementes os nós em losango. Em uma segunda iteração, a árvore de cor laranja foi selecionada para ser removida e uma nova semente com rótulo azul foi inserida. Na Figura 5.3(c), nós em branco pertencem à árvore que foi removida (região laranja), nós em amarelo representam os nós do conjunto de fronteira $\mathcal{F}$ que foram inseridos na fila de prioridades. Além disso temos a nova semente (parte superior, rótulo azul) inserida também em $\mathcal{Q}$. Alguns passos depois no algoritmo $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, ainda na segunda iteração (Figura 5.3(d)), o nó s é avaliado com custo 4 e oferece um custo 12 como extensão de caminho para o nó t . Dado que o custo (12) é maior que o custo atual do nó t , o teste do predecessor é verificado e dado que o teste é verdadeiro o nó t é conquistado pelo caminho $\pi_s^P \cdot \langle s, t \rangle$ (Figura 5.3(e)).

Depois de alguns passos (Figuras 5.3(e-h)) na Figura 5.3(i) o nó u , destacado em vermelho, sai da fila de prioridades antes que o nó t e o conquista, formando um ciclo de predecessores. Além disso, o componente conexo que contém o nó t não tem um nó raiz associado, provocando desse modo uma *região desconexa* das sementes (Figura 5.3(j)). Finalmente na Figura 5.3(k) é mostrado o resultado correto, obtido pela execução do algoritmo $IFT_{(S_2)}$.

Figura 5.3: Exemplo de inconsistência no mapa de rótulos para a função f_{abs_add} . (a) Grafo G , no qual o valor em cada nó é o valor da intensidade na imagem $\hat{I}$. (b) Resultado da $IFT_{(S_1)}$ com duas sementes. (c) Remoção da semente laranja e inserção de uma nova semente com o rótulo azul. Os nós amarelos representam os nós da fronteira e a nova semente, que foram inseridos em $\mathcal{Q}$. (d) O nó s é avaliado e conquista o nó t por meio do teste do predecessor. (e) O nó t tem como predecessor o nó s . (f-h) Alguns passos do algoritmo $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$. Nenhum rótulo consegue conquistar os nós t e u . (i) O nó t sai da fila antes que o seu predecessor (dado que tem um custo menor), então o nó t consegue conquistar o nó s oferecendo um custo menor, gerando um ciclo de predecessores. (j) Resultado da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, apresenta uma região desconexa. (k) Resultado correto obtido pela $IFT_{(S_2)}$.

Em uma aplicação como a geração de superpixels (que será apresentada no Capítulo 7), as inconsistências de rótulo geradas durante o fluxo diferencial com a DIFT* e a função f_D (Equação 5.1)

teriam um efeito conforme ilustrado na Figura 5.4.

Figura 5.4: Resultado da geração de superpixels empregando o algoritmo $DIFT^*$. a) Imagem original. b) Imagem de rótulos obtidas pelo algoritmo $DIFT^*$, note as inconsistências dos rótulos vermelhos. c) Zoom nas regiões desconexas.

5.3 Inconsistências no mapa de custos

O Algoritmo 1 da IFT para funções NMI, pode gerar múltiplos resultados válidos, dependendo da ordem do processamento das sementes, das regiões de empate e da ordem em que os arcos são visitados. O importante é garantir que o resultado do fluxo diferencial

$$\langle IFT_{(\mathcal{S}_1)}, DIFT_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}, \dots, DIFT_{(\Delta_{\mathcal{S}_n}^+, \Delta_{\mathcal{S}_n}^-, c_{n-1})} \rangle$$

corresponde a um resultado possível de ser obtido por meio da $IFT_{(\mathcal{S}_n)}$.

Estudando o comportamento da $DIFT^*$ para funções de custo com aumentos baseados na raiz, além das inconsistências de rótulos é possível encontrar também inconsistências no mapa de custos C .

A Figura 5.5 mostra a inconsistência de custo que é gerada pela $DIFT^*$ com a função de conectividade f_{euc} (Equação 5.2) e uma vizinhança 8. Seja um grafo G , no qual os retângulos não fazem parte do grafo (Figura 5.5(a)), em que é executada uma primeira $IFT_{(\mathcal{S}_1)}$, com uma semente na parte inferior (rótulo verde). Os passos desta iteração são mostrados nas Figuras 5.5(b-d) e o resultado da $IFT_{(\mathcal{S}_1)}$ é mostrado na Figura 5.5(e).

Depois, em uma segunda iteração (execução de $DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$), uma nova semente r_2 é inserida com rótulo azul (Figura 5.5(f)). Nesta mesma iteração, após alguns passos (Figuras 5.5(f-g)), quando o nó s é avaliado (Figura 5.5(h)), este oferece ao nó t um custo de caminho igual a 5, que é maior que o custo atual do nó t . Porém, pelo teste do predecessor, o nó t é conquistado pela extensão do caminho $(\pi_{r_2 \rightsquigarrow s}^P \cdot \langle s, t \rangle)$ como é mostrado na Figura 5.5(i). Como resultado final da execução da $DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$ obtemos a Figura 5.5(j). Comparando os resultados dos algoritmos $DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$ e $IFT_{(\mathcal{S}_2)}$ (Figuras 5.5(j-k)), é possível perceber que o nó t tem seu valor de custo distinto, o valor correto é o valor definido pela execução da $IFT_{(\mathcal{S}_2)}$.

Figura 5.5: *DIFT** com uma relação de adjacência de vizinhança 8 e a função f_{euc} (Equação 5.2). (a) Grafo G no qual os retângulos não fazem parte do grafo. (b-d) Inserção de uma semente com o rótulo verde e sua propagação no grafo. (e) Resultado da $IFT_{(S_1)}$. (f-g) Inserção de uma nova semente em $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ (rótulo azul), e sua propagação. (h-i) Na execução do Algoritmo $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, o nó s oferece um custo de 5 ao nó t , dado que o custo é maior, t é conquistado por meio do teste do predecessor. (j) Resultado da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$. (k) Resultado correto por meio do Algoritmo $IFT_{(S_2)}$ para o mesmo conjunto de sementes, note a diferença de custo do nó t .

Capítulo 6

Algoritmo DIFTmod

Neste capítulo é apresentado o novo algoritmo DIFTmod, que consegue uma solução de alto desempenho, capaz de lidar com os problemas de inconsistências dos mapas relatados na Seção 4.5.1 e no Capítulo 5. Serão apresentadas a análise e as soluções para cada problema, para ao final convergir num único algoritmo capaz de lidar com funções clássicas MI e funções de custo com aumento baseado na raiz. O algoritmo proposto obtém um resultado válido da IFT, ou seja, como a IFT pode ter vários resultados possíveis, o resultado do fluxo diferencial via DIFTmod corresponde a um deles.

6.1 Problema na inclusão do teste de estados

Análise do problema Com o teste de estados, o algoritmo DIFT* produz uma inconsistência de rótulos tal como mencionado na Seção 4.5.1. A inconsistência produzida depende da escolha da política de desempate na fila de prioridades $\mathcal{Q}$.

Na Figura 6.1 é mostrado um exemplo de inconsistência de rótulos usando a função de custo f_{max} (Equação 2.3) da classe MI, vizinhança 4 e política de desempate FIFO (exemplo da Seção 4.5.1). Na Figura 6.1(b) é mostrado o resultado da primeira $IFT_{(\mathcal{S}_1)}$ com duas sementes sobre o grafo $\mathbf{G}$ da Figura 6.1(a). Na seguinte iteração ($DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$) o conjunto de sementes é modificado com a inserção de uma nova semente r_3 e a remoção da região laranja (semente r_2) (Figura 6.1(b)). Note que a nova semente foi marcada no interior da região com raiz em r_1 . Durante a execução da $DIFT^*_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$, é criado o conjunto de fronteira $\mathcal{F}$ e os nós $s \in \mathcal{F}$ são inseridos na fila de prioridades antes que a nova semente r_3 (Figura 6.1(c)). A inconsistência de rótulos ocorre se existe algum nó t , antecessor do nó $s \in \mathcal{F}$ no caminho $\pi_{r_1 \rightsquigarrow s}^{P_{i-1}}$, que passa a ser conquistado na i -ésima iteração, por um caminho $\pi_{r_3 \rightsquigarrow t}^{P_i}$ com raiz na nova semente r_3 , tendo o mesmo custo que $\pi_{r_1 \rightsquigarrow s}^{P_{i-1}}$ (isto é $f(\pi_{r_1 \rightsquigarrow s}^{P_{i-1}}) = f(\pi_{r_3 \rightsquigarrow t}^{P_i})$) (Figura 6.1(d)). Se isso acontecer, o nó s será avaliado antes que o nó t (dado que foi inserido primeiramente em $\mathcal{Q}$) e o estado do nó s será alterado para “preto” (Figura 6.1(e)), inviabilizando qualquer atualização posterior do nó s . Como a função f_{max} é não decrescente, temos que $f(\pi_{r_1 \rightsquigarrow t}^{P_{i-1}}) \leq f(\pi_{r_1 \rightsquigarrow s}^{P_{i-1}})$. Logo a única forma do nó t ter sido conquistado por um caminho com origem em r_3 e com o custo apontado se dá pelo teste do predecessor. Em outras palavras, por ser f_{max} uma função MI, quando o nó t , que é o antecessor do nó s (Figura 6.1(f)), é avaliado, o caminho estendido $\pi_{r_3 \rightsquigarrow t}^P \cdot \langle t, s \rangle$ vai oferecer um custo igual ao custo atual do nó s , mas como o nó s já tinha sido avaliado (cor preto), o caminho com origem no r_3 não pode atualizar os atributos do nó s . Nessas circunstâncias, os caminhos estendidos a partir da subárvore com origem

em s propagarão um rótulo incorreto gerando assim a “inconsistência de rótulos” (Figura 6.1(f-h)). Sem o teste de estados, o nó s poderia ser atualizado, mas também seria atualizada toda a subárvore com origem no nó s . Ou seja, cada nó poderia ser atualizado mais de uma vez, durante a mesma iteração da $DIFT_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, comprometendo o tempo de execução.

Figura 6.1: Inconsistência no mapa de rótulos na $DIFT^*$ para a função $f_{\max}$ (Equação 2.3), política de desempate FIFO e vizinhança 4. (a) Grafo derivado de uma imagem. (b) Resultado da $IFT_{(S_1)}$ com duas sementes (nós sementes em forma de losango). Além disso, para a $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ temos a remoção do rótulo laranja e inserção de uma nova semente (rótulo azul) no canto superior esquerdo (c) O conjunto $\mathcal{F}$ é criado e seus nós são inseridos na fila de prioridades $\mathcal{Q}$, acompanhados da nova semente (rótulo azul). Os nós com a cor amarela representam vértices que estão na fila de prioridades. (d) O nó t é conquistado com o mesmo custo $f_{\max}(\pi_{r_3 \rightsquigarrow t}) = f_{\max}(\pi_{r_1 \rightsquigarrow t})$, mas por outra raiz. (e) O nó s é avaliado e muda seu estado para “preto”. (f) O nó t não pode conquistar o nó s (mesmo sendo seu predecessor), s e t não têm o mesmo rótulo embora eles possuam a mesma raiz. (g) Propagação da inconsistência. (h) Resultado da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$. Note as inconsistências nos rótulos. (i) Resultado da $IFT_{(S_2)}$ com os mapas de raízes e de rótulos corretos.

Moya e Falcão foram os primeiros a identificar este problema na DIFT*.

Em [Moy15], foram analisadas as seguintes alternativas de solução para a inconsistência no mapa de rótulos da DIFT*, com funções MI:

1. Definir a política de desempate da fila de prioridades $\mathcal{Q}$ apenas como LIFO.
2. Tratar uma operação conjunta de adição e remoção como 2 operações independentes.
3. Remover o teste de estados da DIFT* (Algoritmo 4).

Em uma política LIFO todos os nós a serem conquistados com o mesmo custo seriam atribuídos à nova semente, no entanto, como indicado na Seção 3.3.2 uma política LIFO não gera uma segmentação equitativa bem balanceada, o que não é desejável na prática. Para tratar uma operação conjunta de adição e remoção como duas operações independentes, seria necessário executar duas vezes o laço principal da DIFT* (Algoritmo 4), acrescentando esforço computacional. A terceira solução corresponde a usar o Algoritmo 2 da DIFT original, que claramente não suporta funções NMI e que pode atualizar cada nó mais de uma vez, conforme mencionado anteriormente.

No fluxo diferencial em alguns poucos casos, a política FIFO pode gerar uma segmentação com área de desempate desbalanceada, estes resultados dependem da semente removida e do grafo inicial. Não obstante estes resultados são preferíveis aos resultados obtidos por meio de uma política LIFO.

6.1.1 Proposta de solução para as inconsistências geradas pela inclusão do teste de estados, com funções MI

Nesta Seção é proposto um primeiro algoritmo chamado DIFTmod^{FMI} (Algoritmo 5), para solucionar as inconsistências no mapa de rótulos L , mantendo o teste de estados e uma política de desempate FIFO. Para evitar as possíveis inconsistências são adicionadas algumas condições para prevenir que nós de fronteira sejam avaliados antes que um nó antecessor.

Seja r_3 a nova semente inserida, s um nó que pertence ao conjunto de fronteira $\mathcal{F}$ e t um nó antecessor de s . É avaliado se existe um caminho $\pi_{r_3 \rightsquigarrow u}^P \cdot \langle u, t \rangle$ com origem na raiz r_3 , tal que:

1. O custo oferecido ao nó t é igual ao custo anterior em $C(t)$, e
2. O nó u é o predecessor do nó t , isto é, $P(t) = u$, e
3. A raiz de u é diferente da raiz do nó t , isto é, $R(u) \neq R(t)$.

Se as condições anteriores forem verdadeiras, então o nó t deve ser inserido na fila de prioridades (Figura 6.2(a)), empregando uma política de desempate LIFO (Figura 6.2(b)). Ou seja o nó t deve ser inserido no início da fila de prioridades para o custo $C(t)$. Logo que o nó t for inserido, a fila de prioridades deve retornar a sua política original FIFO (Figura 6.2(c)).

Figura 6.2: Inserção de nós na fila de prioridades $\mathcal{Q}$. (a) As setas pretas apontam para os buckets/posições de mais alta prioridade de remoção. (b) Se o nó t é inserido no bucket de custo 8 como LIFO, ele vai ser inserido no início (topo) da lista de custo 8 e será avaliado antes que os demais nós destacados em amarelo. (c) Se o nó t' é inserido com política FIFO, ele é inserido no final da lista de custo 8 e será avaliado depois que os nós destacados em amarelo.

Na Figura 6.3 são apresentados graficamente os passos do algoritmo DIFTmod^{FMI} , na Figura 6.3(a) é mostrado o grafo $\mathcal{G}$ derivado da imagem $\hat{I}$. A Figura 6.3(b) representa a remoção da raiz r_2 e a inserção da nova semente r_3 sobre o resultado da $\text{IFT}_{(\mathcal{S}_1)}$. Durante a segunda iteração da $\text{DIFTmod}^{FMI}_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, c_1)}$, o conjunto de fronteira é criado e os nós que pertencem a $\mathcal{F}$ são inseridos na fila de prioridades (Figura 6.3(c)). Note que o nó s é inserido antes que a nova semente r_3 . Os nós amarelos representam os vértices que pertencem à fila de prioridades $\mathcal{Q}$. Quando o nó t for visitado pela extensão de caminho $(\pi_{r_3 \rightsquigarrow u}^P \cdot \langle u, t \rangle)$ com raiz na nova semente (Figura 6.3(e)), serão avaliadas as novas condições. Como todas são verdadeiras, o nó t será inserido na fila de prioridades com uma política LIFO, garantindo deste modo que t sairá da fila $\mathcal{Q}$ antes que o nó s (Figura 6.3(f)). Assim é possível obter um mapa de rótulos L consistente (Figura 6.3(g)). O resultado da $\text{IFT}_{(\mathcal{S}_2)}$ é apresentado na Figura 6.3(h).

Figura 6.3: Representação do algoritmo $DIFTmod^{FMI}$, com a função f_{max} (Equação 2.3) e vizinhança 4. (a) Grafo derivado de uma imagem. (b) Resultado da $IFT_{(S_1)}$, adição da semente r_3 (rótulo azul) e remoção da região laranja. (c) Os nós amarelos representam a nova semente r_3 e o conjunto de fronteira, inseridos em $\mathcal{Q}$. (d) Propagação do rótulo de r_3 . (e) Instante em que o nó t é conquistado pela semente r_3 e as condições da Seção 6.1.1 são avaliadas. Dado que todas são verdadeiras o nó t é inserido com uma política LIFO na fila de prioridades. (f) O nó t é avaliado antes que o nó s , deixando os mapas de raízes e de rótulos coerentes, assim como a árvore toda. (g) Resultado da $DIFTmod^{FMI}_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$. (h) Resultado da $IFT_{(S_2)}$.

A seguir é apresentado o algoritmo da DIFTmod^{FMI}.

Algoritmo 5: Algoritmo DIFTmod^{FMI}

Entrada : Imagem $\hat{I}$, relação de adjacência $\mathcal{A}$, o conjunto de sementes a adicionar Δ_S^+ , conjunto de sementes a remover Δ_S^- , função de custo monotonicamente incremental f , os mapas P, L, R, C e a função de rotulação λ .
Saída : Mapas P, L, R, C .
Auxiliares: Fila de prioridades $\mathcal{Q}$ e conjunto de nós de fronteira $\mathcal{F}$.

```

1  $\mathcal{Q} \leftarrow \emptyset$ 
2 para cada  $p \in \mathcal{I}$  faça  $estado(p) \leftarrow branco$ 
3 se  $\Delta_S^- \neq \emptyset$  então
4    $(P, L, R, C, \mathcal{F}) \leftarrow \text{DIFT-LimpaÁrvores}(P, L, R, C, \mathcal{A}, \Delta_S^-)$ 
5    $\mathcal{F} \leftarrow \mathcal{F} \setminus \Delta_S^+$ 
6   para cada  $s \in \mathcal{F}$  faça
7      $\lfloor$  Insira  $s$  em  $\mathcal{Q}$  com custo  $C(s)$ 
8 para cada  $s \in \Delta_S^+$  faça
9   se  $f(\langle s \rangle) < C(s)$  então
10     $P(s) \leftarrow nil, L(s) \leftarrow \lambda(s), R(s) = s$ 
11     $C(s) \leftarrow f(\langle s \rangle)$  e insira  $s$  em  $\mathcal{Q}$  com custo  $C(s)$ 
12 enquanto  $\mathcal{Q} \neq \emptyset$  faça
13   Remova  $s$  de  $\mathcal{Q}$  tal que  $C(s)$  seja mínimo
14    $estado(s) \leftarrow preto$ 
15   para cada  $t \in \mathcal{A}(s)$  faça
16     se  $estado(t) \neq preto$  então
17       Calcule  $tmp \leftarrow f(\pi_s^P \cdot \langle s, t \rangle)$ 
18       se  $tmp = C(t)$  e  $P(t) = s$  e  $R(t) \neq R(s)$  então
19         se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
20          $L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
21         Insira  $t$  em  $\mathcal{Q}$  com política LIFO
22       senão
23         se  $tmp < C(t)$  então
24           se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
25            $P(t) \leftarrow s, L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
26            $C(t) \leftarrow tmp$  e insira  $t$  em  $\mathcal{Q}$  com política FIFO

```

No Algoritmo 5, adicionamos os novos testes (Linha 18) para evitar que nós previamente inseridos, pertencentes ao conjunto $\mathcal{F}$, saiam da fila de prioridades antes do que algum nó antecessor (com o mesmo custo). As Linhas 23-26, aplicam o procedimento padrão do algoritmo da IFT.

6.2 Solução para funções de custo com aumentos baseados na raiz

As funções apresentadas f_{euc} e f_{abs_add} têm como propriedade, que o aumento de custo Δ_C oferecido na extensão do caminho $\pi_{r' \rightsquigarrow s} \cdot \langle s, t \rangle$ é influenciado pelas propriedades da raiz.

Embora as modificações da Seção 6.1 evitam os problemas de inconsistências de rótulos e de raízes, gerados ao manter o *teste de estados* para funções MI, estas não resolvem os problemas da Seção 5.1, para funções de custo do caminho com aumentos Δ_C baseados na raiz.

No algoritmo da DIFT*, dado o cálculo de um novo caminho estendido $(\pi_s^P \cdot \langle s, t \rangle)$, este só é considerado se o custo computado for menor que o custo atual de t no mapa C , ou se o nó s é predecessor do nó t , isto é, $P(t) = s$. Mas nenhum algoritmo visto até agora inclui condições caso o novo custo seja maior que o custo atual de t . Então, como foi mencionado na Seção 5.2, as inconsistências para funções com aumentos baseados na raiz, acontecem quando o teste do predecessor for verdadeiro e o valor computado pela extensão de caminho for maior que o custo atual ($f(\pi_s^P \cdot \langle s, t \rangle) > C(t)$). É por isso que o novo algoritmo DIFTmod adiciona as condições necessárias para avaliar este caso, Note que a relação custo maior e predecessor é conjuntiva, ou seja as duas condições devem ser verdadeiras simultaneamente.

Na Figura 6.4, recapitulamos a origem da inconsistência com o algoritmo DIFT*. Dado um grafo G com o valor das intensidades nos nós (Figura 6.4(a)), a função de conexidade f_{abs_add} (Equação 5.3) e a vizinhança 4, foram inseridas duas sementes com rótulos verde e laranja, o resultado da $IFT_{(S_1)}$ é apresentado na Figura 6.4(b). Para a segunda iteração da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ é inserida uma nova semente r_3 (com rótulo azul) e a região laranja é selecionada para ser removida (Figura 6.4(b)). Durante a execução da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, os nós do conjunto de fronteira e a nova semente r_3 são inseridos na fila de prioridades, sendo que nós amarelos representam os nós em Q (Figura 6.4(c)). Posteriormente, quando o nó s é avaliado (Figura 6.4(d)), o valor da extensão de caminho de s para t é maior que $C(t)$, isto é, $f(\pi_{r_3 \rightsquigarrow s} \cdot \langle s, t \rangle) > C(t)$, mas como s é o predecessor do nó t , t será conquistado pelo caminho com origem na semente r_3 . É assim que a subárvore que deriva do nó t fica em um estado predisposto a originar inconsistências. As inconsistências podem acontecer, se existir algum nó u que pertence à subárvore que deriva do nó t , com um valor em C menor que o custo oferecido ao nó t (Figura 6.4(d)). O nó u sairá antes que seu antecessor que se encontra na fila de prioridades (Figura 6.4(e)), propagando um rótulo errado e criando regiões desconexas (Figura 6.4(f)).

Figura 6.4: Inconsistência de rótulos para a função NMI f_{abs_add} (Equação 5.3) e vizinhança 4. (a) Grafo com o valor das intensidades nos nós. (b) Resultado da $IFT_{(S_1)}$ com duas sementes. A região laranja é selecionada para remoção e uma nova semente é inserida. (c) Na execução da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, o conjunto de fronteira e a nova semente são inseridos em Q . (d) O nó t é conquistado com o teste do predecessor pelo caminho com origem na semente r_3 (com um custo maior). (e) O nó u , que pertence à subárvore com origem em t , é avaliado antes que seu antecessor, gerando inconsistências. (f) Resultado da $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$

Dadas as circunstâncias anteriores, a solução considerada é prevenir que nenhuma subárvore com origem no nó t (que fora conquistado com um custo maior) possa ficar em um estado predisposto a originar inconsistências. Por isso adicionamos as seguintes condições:

- Se o custo oferecido pela extensão de caminho for maior $f(\pi_{r \rightsquigarrow s}^P \cdot \langle s, t \rangle) > C(t)$, e o nó s é predecessor do nó t , isto é, $P(t) = s$. Então:
 - Descartar a subárvore que tem origem no nó t , colocando valores de custo infinito nos seus nós.
 - Transformar em nós de fronteira todos os vizinhos da subárvore e inseri-los na fila de prioridades.

Na exemplo da Figura 6.5 são apresentadas graficamente as novas condições da DIFTmod (Algoritmo 6). Dados um grafo G com o valor das intensidades nos nós (Figura 6.5(a)), a função de conexidade f_{abs_add} (Equação 5.3) e a vizinhança 4, foi executada a primeira $IFT_{(S_1)}$ com duas sementes. Na segunda iteração ($DIFTmod_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$) os conjuntos Δ_S^+ e Δ_S^- foram modificados. Na Figura 6.5(c) os nós de fronteira e a nova semente são representados em nós amarelos e estão em $\mathcal{Q}$. Na Figura 6.5(d) o nó s oferece um custo maior ao nó t e s é o predecessor do nó t , como as duas condições são verdadeiras a subárvore com a origem no nó t é descartada (Figura 6.5(d)), deixando seus nós como árvores triviais de custo infinito (nós vermelhos) e seus nós vizinhos são transformados em nós de fronteira (nós tracejados), que são inseridos na fila de prioridades com seus atributos atuais. A subárvore removida será conquistada posteriormente pelos nós em $\mathcal{Q}$. As Figuras 6.5(f-o) mostram os passos após as modificações.

Figura 6.5: parte I Representação gráfica da proposta de solução para funções NMI. Exemplo usando a função f_{abs_add} (Equação 5.3) e vizinhança 4. (a) Grafo com o valor das intensidades nos nós. (b) Resultado da $IFT_{(S_1)}$ com duas sementes. A região laranja é seleccionada para remoção e uma nova semente é inserida. (c) Na execução da $DIFTmod_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ o conjunto de fronteira e a nova semente são inseridos em $\mathcal{Q}$. (d) As novas condições são avaliadas, o custo oferecido de s para t é maior que o custo $C(t)$ e s é predecessor do t . (e) Como as novas condições são verdadeiras, a subárvore com a origem no nó t é descartada (nós vermelhos) e seus nós vizinhos são transformados em nós de fronteira (nós tracejados). (f-o) Passos após a remoção da subárvore, todos os nós são conquistados com o rótulo correto.

Figura 6.5: continuação Representação gráfica da proposta de solução para funções NMI. Exemplo usando a função f_{abs_add} (Equação 5.3) e vizinhança 4. (g-o) Passos após a remoção da subárvore, todos os nós são conquistados a partir dos nós inseridos em $\mathcal{Q}$.

Na Figura 6.6 são apresentados os resultados dos fluxos sequencial e diferencial para as duas interações do exemplo anterior, aplicando a função f_{abs_add} (Equação 5.3). Note que não existem inconsistências no resultado obtido pelo algoritmo DIFTmod.

Figura 6.6: Resultados obtidos pelos algoritmos. Note que a $DIFTmod_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ consegue um resultado idêntico ao resultado da $IFT_{(S_2)}$

Com a intenção de apresentar um único algoritmo, as soluções da Subseção 6.1.1 e da Seção 6.2 são unificadas no algoritmo DIFTmod (Algoritmo 6) apresentado a seguir.

Algoritmo 6: Algoritmo DIFTmod

Entrada : Imagem $\hat{I}$, relação de adjacência $\mathcal{A}$, o conjunto de sementes a adicionar Δ_S^+ , conjunto de sementes a remover Δ_S^- , função de custo f , os mapas P, L, R, C e a função de rotulação λ .

Saída : Mapas P, L, R, C .

Auxiliares: Fila de prioridades $\mathcal{Q}$ e conjunto de nós de fronteira $\mathcal{F}$.

```

1  $\mathcal{Q} \leftarrow \emptyset$ 
2 para cada  $p \in \mathcal{I}$  faça  $estado(p) \leftarrow branco$ 
3 se  $\Delta_S^- \neq \emptyset$  então
4    $(P, L, R, C, \mathcal{F}) \leftarrow \text{DIFT-LimpaÁrvores}(P, L, R, C, \mathcal{A}, \Delta_S^-)$ 
5    $\mathcal{F} \leftarrow \mathcal{F} \setminus \Delta_S^+$ 
6   para cada  $s \in \mathcal{F}$  faça
7      $\text{Insira } s \text{ em } \mathcal{Q} \text{ com custo } C(s)$ 
8 para cada  $s \in \Delta_S^+$  faça
9   se  $f(\langle s \rangle) < C(s)$  então
10     $P(s) \leftarrow nil, L(s) \leftarrow \lambda(s), R(s) = s$ 
11     $C(s) \leftarrow f(\langle s \rangle)$  e insira  $s$  em  $\mathcal{Q}$  com custo  $C(s)$ 
12 enquanto  $\mathcal{Q} \neq \emptyset$  faça
13   Remova  $s$  de  $\mathcal{Q}$  tal que  $C(s)$  seja mínimo
14    $estado(s) \leftarrow preto$ 
15   para cada  $t \in \mathcal{A}(s)$  faça
16     se  $estado(t) \neq preto$  então
17       Calcule  $tmp \leftarrow f(\pi_s^P \cdot \langle s, t \rangle)$ 
18       se  $tmp > C(t)$  e  $P(t) = s$  então
19          $\text{DIFT-PodaSubÁrvore}(P, L, R, C, \mathcal{Q}, estado, t)$ 
20       senão
21         se  $tmp = C(t)$  e  $P(t) = s$  e  $R(t) \neq R(s)$  então
22           se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
23            $L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
24           Insira  $t$  em  $\mathcal{Q}$  com política LIFO
25         senão
26           se  $tmp < C(t)$  então
27             se  $t \in \mathcal{Q}$  então remova  $t$  de  $\mathcal{Q}$ 
28              $P(t) \leftarrow s, L(t) \leftarrow L(s), R(t) \leftarrow R(s)$ 
29              $C(t) \leftarrow tmp$  e insira  $t$  em  $\mathcal{Q}$  com política FIFO

```

No Algoritmo 6 as Linhas 18-19 avaliam as condições para realizar a poda da subárvore pelo Algoritmo 7, a Linha 19 é executada unicamente se o custo oferecido pelo caminho $(\pi_{r \rightsquigarrow s}^P \cdot \langle s, t \rangle)$ for maior que o custo $C(t)$ e o nó s é o predecessor do nó t .

As Linhas 21-24 correspondem à solução da Subseção 6.1.1 e as Linhas 26-29 são o procedimento padrão da IFT.

6.2.1 Algoritmo de poda

O algoritmo DIFT-PodaSubÁrvore (Algoritmo 7), possui uma fila de nós T e o conjunto K , que processam os nós da subárvore a ser removida.

Algoritmo 7: Algoritmo DIFT-PodaSubÁrvore

Entrada : Mapas P , L , R , C , fila de prioridades $\mathcal{Q}$, array *estado* e o nó t (raiz da subárvore que será removida).

Saída : Mapas P , L , R , C e a fila de prioridades $\mathcal{Q}$.

Auxiliares: Fila T e o conjunto K .

```

1   $T \leftarrow \emptyset, K \leftarrow \emptyset$ 
2  Insira  $t$  na fila  $T$ 
3  enquanto  $T \neq \emptyset$  faça
4      Remova  $p$  de  $T$ 
5       $estado(p) \leftarrow branco$ 
6       $P(p) \leftarrow nil, L(p) \leftarrow nil, R(p) \leftarrow nil$ 
7       $C(p) \leftarrow +\infty$ 
8      se  $p \in \mathcal{Q}$  então remova  $p$  de  $\mathcal{Q}$ 
9      para cada  $q \in \mathcal{A}(p)$  faça
10         se  $p = P(q)$  então
11             Insira  $q$  na fila  $T$ 
12         else
13             se  $q \notin K$  e  $C(q) \neq +\infty$  então
14                 Insira  $q$  em  $K$ 
15 enquanto  $K \neq \emptyset$  faça
16     Remova  $p$  de  $K$ 
17     se  $C(p) \neq +\infty$  e  $p \notin \mathcal{Q}$  então
18          $estado(p) \leftarrow branco$ 
19         Insira  $p$  em  $\mathcal{Q}$  com custo  $C(p)$ 

```

No Algoritmo 7 a subárvore a ser removida é visitada em um percurso em largura, armazenando seus nós na fila T para logo inserir os vizinhos da subárvore com a origem no nó t na fila de prioridade.

Na Linha 2 o nó t , que é a raiz da subárvore a ser removida, é inserido na fila T . Nas Linhas 5-7 os nós que pertencem à subárvore são inicializados como árvores triviais de custo infinito. Durante o percurso na subárvore (Linhas 9-11), se o nó p é predecessor do nó q , q é inserido na fila T para assim continuar o processamento da subárvore. Se p não é predecessor de q (Linhas 13-14) o nó q é inserido no conjunto K .

Após do laço das Linhas 3-14 o conjunto K contém os nós vizinhos da subárvore removida. Nas Linhas 15-19, se o custo do nó avaliado é diferente do valor infinito, ou seja, não é uma árvore trivial então ele deve se tornar um nó de fronteira. Finalmente os nós de fronteira da subárvore são inseridos na fila de prioridades $\mathcal{Q}$ e seus *estados* são transformados para o estado *branco*.

O problema apresentado na Seção 5.3 (inconsistência no mapa de custos), também é solucionado pelo algoritmo DIFTmod. A Figura 6.7 representa a execução do algoritmo DIFTmod com a função f_{euc} (Equação 5.2).

Na Figura 6.7(a) é representado um grafo G , com vizinhança 8. O resultado da $IFT_{(S_1)}$ com uma semente é mostrado na Figura 6.7(b). Para a segunda iteração da $DIFTmod_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ uma nova semente (com rótulo azul) é inserida (Figura 6.7(c)). Durante a execução do algoritmo (Figuras 6.7(d-e)), o nó s oferece um custo maior (5) ao nó t (Figura 6.7(e)). Dado que o custo é maior e s é predecessor do nó t as novas condições da DIFTmod são verdadeiras, portanto a subárvore com a origem no nó t é removida (Figura 6.7(f)) e os nós vizinhos são inseridos na fila de prioridades como nós de fronteira (nós tracejados). As Figuras 6.7(g-i) mostram os passos posteriores da execução.

Figura 6.7: Execução do algoritmo DIFTmod para a função f_{euc} . (a) Representa um grafo com vizinhança 8. (b) Resultado da $IFT_{(S_1)}$ com uma única semente. (c) Uma nova semente com o rótulo azul é inserida. (d) Propagação do rótulo azul. (e) O custo oferecido pelo nó s para o nó t é maior que o custo $C(t)$, além disso s é o predecessor do nó t . (f) Dado que as condições são verdadeiras, a subárvore com a origem no nó t é removida e os vizinhos da subárvore são inseridos em $\mathcal{Q}$. (g-i) Passos posteriores da execução da $DIFTmod_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$.

A Figura 6.8 mostra os resultados da execução dos algoritmos $IFT_{(S_2)}$, $DIFT^*_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$ e $DIFTmod_{(\Delta_{S_2}^+, \Delta_{S_2}^-, c_1)}$, para a função f_{euc} (Equação 5.2). Note que não existem inconsistências no resultado obtido pelo algoritmo DIFTmod.

Figura 6.8: Resultados obtidos pelos algoritmos. Note que a $DIFTmod_{(\Delta_{s_2}^+, \Delta_{s_2}^-, c_1)}$ consegue um resultado idêntico ao resultado da $IFT_{(s_2)}$

Capítulo 7

DIFTmod para a aplicação de geração de superpixels

Neste capítulo são apresentados os resultados do trabalho, aplicados ao método de geração de superpixels IFT-SLIC [ACFM15].

7.1 Algoritmo IFT-SLIC

Os superpixels são o resultado de uma super-segmentação não supervisionada de uma imagem [RFSR13] e têm sido utilizados em uma ampla variedade de aplicações (rastreamento de objetos em vídeo [WLYY11], segmentação de imagens [WLG⁺17], detecção de objetos [SDS13], reconstrução 3D [TC17]). Um dos métodos mais conhecidos para sua geração é o algoritmo *Simple Linear Iterative Clustering* (SLIC) [ASS⁺12]. O SLIC amostra k clusters iniciais¹ sobre uma grade regularmente espaçada de S em S pixels ($S = \sqrt{N/k}$, em que N é o número de pixels da imagem), e atualiza cada região por meio de clusters adaptativos k-means. O Algoritmo IFT-SLIC [ACFM15] considera a inicialização de k sementes iniciais de forma semelhante ao SLIC e adapta os superpixels/regiões calculando uma IFT com a função de conectividade f_{iftslc} (Equação 7.1)², segmentando a imagem em k regiões. Esta etapa é repetida um número n de vezes, em uma cadeia de IFTs, atualizando em cada uma das iterações as posições das sementes. O parâmetro *alfa* α , controla a compacidade do superpixel, sendo que quanto menor o valor de alfa mais compacto se torna o superpixel. Na Figura 7.1 é mostrado o resultado do cálculo de superpixels por meio da IFT-SLIC com os parâmetros $k = 800$ e $\alpha = 0.09$.

$$\begin{aligned} f_D(\pi_s = \langle s \rangle) &= \begin{cases} 0 & , \text{ se } s \in \mathcal{S} \\ +\infty & , \text{ caso contrário} \end{cases} \\ f_D(\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle) &= f_D(\pi_{r \rightsquigarrow s}) + (\|I(r) - I(t)\| \cdot \alpha)^\beta + \|t - s\| \end{aligned} \quad (7.1)$$

¹ SLIC recebe o parâmetro k' que é o número desejado de superpixels, mas, não garante este número exato de superpixels. Unicamente $k \approx k'$ superpixels são empregados na malha regularmente espaçada [ASS⁺12].

²As cores I são calculadas no espaço CIELAB.

Figura 7.1: Superpixels gerados pelo algoritmo IFT-SLIC com $k = 800$ e $\alpha = 0.09$ numa imagem de 154401 pixels.

Atualização das sementes

Seja $c_i^m = [l_i^m \ a_i^m \ b_i^m \ x_i^m \ y_i^m]$ o centroide do superpixel i na iteração m , os valores $[l^m, a^m, b^m]$ representam o vetor da cor média do superpixel e (x^m, y^m) representa a coordenada espacial da raiz. A raiz do superpixel é atualizada se:

$$\| [l_i^m \ a_i^m \ b_i^m] - [l_i^{m+1} \ a_i^{m+1} \ b_i^{m+1}] \| > \epsilon_c$$

ou se

$$\| [x_i^m \ y_i^m] - [x_i^{m+1} \ y_i^{m+1}] \| > \epsilon_s$$

para os parâmetros ϵ_c e ϵ_s definidos pelo usuário. Por exemplo, a Figura 7.2 mostra a atualização das raízes dos superpixels, para um valor de k igual a 400, a Figura 7.2(a) representa a posição inicial das raízes em uma malha regularmente espaçada, as Figuras 7.2(b-d) mostram a movimentação das raízes, na primeira, quinta e décima iteração da cadeia de IFTs, $\langle IFT_{(S_1)}, IFT_{(S_2)}, \dots, IFT_{(S_n)} \rangle$ para $n = 10$ (IFTx10). O resultado da IFTx10 é exposto na Figura 7.2(e) e na Figura 7.2(f) é mostrada a imagem de superpixels com a cor média em cada superpixel.

Figura 7.2: (a) Posição inicial das raízes. (b-d) Atualização de raízes dos superpixels nas iterações 1, 5 e 10. (e) Resultado da IFTx10. (f) Imagem de superpixels com a cor média em cada um deles.

7.2 Modo diferencial da IFT-SLIC

Dado que o cálculo da IFT-SLIC é uma cadeia de execuções de IFTs, é possível usar o modo diferencial para reduzir o esforço computacional em cada uma das iterações. O Algoritmo 8 mostra o pseudo-código para a geração de superpixels baseado no Algoritmo IFT-SLIC [ACFM15].

Algoritmo 8: Algoritmo diferencial para a geração de superpixels

- Entrada** : Imagem $\hat{I}$, relação de adjacência $\mathcal{A}$ (vizinhança 4), função de custo f_{iftslc} .
Saída : Mapa de rótulos L .
- 1 Converter a imagem $\hat{I}$ para o espaço de cores CIELAB.
 - 2 Calcular o conjunto inicial de raízes $\mathcal{S}_1$, baseado em uma malha regularmente espaçada [ASS⁺12].
 - 3 Rodar uma primeira IFT com o conjunto $\mathcal{S}_1$, a função f_{iftslc} (Equação 7.1) e a vizinhança 4.
 - 4 **para cada** iteração $i \in [2...n]$ **faça**
 - 5 Computar o conjunto $\mathcal{W}$ de superpixels, que devem ser recalculados/atualizados segundo a Subseção 7.1 e adicionar a raiz deles no conjunto $\Delta_{\mathcal{S}_i}^-$
 - 6 **para cada** $superpixel_i \in \mathcal{W}$ **faça**
 - 7 Computar a nova posição da raiz do superpixel_{*i*} e adicioná-la ao conjunto $\Delta_{\mathcal{S}_i}^+$
 - 8 Rodar o *algoritmo diferencial*, sendo $\Delta_{\mathcal{S}_i}^+$ o conjunto de raízes novas, $\Delta_{\mathcal{S}_i}^-$ o conjunto de raízes a remover e $\mathcal{C}_{i-1}$ a cena da execução anterior.
-

Se na Linha 8 do Algoritmo 8 é usado o algoritmo da DIFT*, ocorrem inconsistências no mapa de rótulos, ou seja, regiões com rótulos diferentes das suas raízes (pixels pretos nas Tabelas 7.1 e 7.2). Isto se deve ao fato de que a função de conectividade usada é uma variação da função apresentada na Seção 5.1 (Equação 5.3) e portanto, apresenta os problemas do Capítulo 5.

Tabela 7.1: Regiões desconexas originadas pela DIFT* no Algoritmo IFT-SLIC para $k = 400$, $n = 10$, $\beta = 12$, $\epsilon_c = 5$, $\epsilon_s = 2$ e distintos valores de α .

Tabela 7.2: Regiões desconexas originadas pela *DIFT** no Algoritmo *IFT-SLIC* para $k = 800$, $n = 10$, $\beta = 12$, $\epsilon_c = 5$, $\epsilon_s = 2$ e distintos valores de α .

7.3 DIFTmod para a geração de superpixels

Com o algoritmo proposto DIFTmod, é possível gerar um resultado consistente, capaz de reproduzir o resultado do cálculo dos superpixels da IFT executada 10 vezes (IFTx10). O procedimento segue os passos do Algoritmo 8 utilizando o algoritmo DIFTmod para o cálculo diferencial.

A Figura 7.3 ilustra as etapas do Algoritmo 8, o bloco azul representa a execução da primeira IFT. Em seguida, sobre o resultado da IFT são avaliadas as condições de atualização de raízes da Subsubseção 7.1 (bloco laranja). Se um superpixel tem sua raiz modificada a raiz antiga é inserida no conjunto Δ_S^- (círculos vermelhos) e a raiz nova no conjunto Δ_S^+ (círculos verdes). Os conjuntos de raízes Δ_S^+ e Δ_S^- são os parâmetros para a execução do algoritmo DIFTmod (bloco verde). A execução das etapas (atualização e DIFTmod) é repetida n vezes (nos experimentos deste trabalho 9 iterações foram realizadas)³.

Figura 7.3: Processo diferencial de geração de superpixels. O bloco azul representa a execução da primeira IFT. No bloco laranja é mostrado o teste de atualização para um superpixel (com raiz em r), dado que a raiz r deve ser modificada a semente nova r' é inserida no conjunto Δ_S^+ e a raiz r no conjunto Δ_S^- . Estes conjuntos são os parâmetros para a execução da DIFTmod (bloco verde). As etapas de atualização de raízes e execução da DIFTmod são repetidas n vezes.

7.4 Metodologia dos experimentos

Os experimentos foram conduzidos com o objetivo de comparar o resultado do algoritmo IFT-SLIC calculado de modo diferencial e o resultado da IFT-SLIC_{IFT} que executa 10 iterações sequenciais da IFT.

³São feitas no total 10 execuções (1 execução da IFT + 9 execuções diferenciais) assim como em [ACFM15].

Correspondência de rótulos Seja L_1 (Figura 7.5(a)) o resultado do cálculo de superpixels do algoritmo diferencial (DIFT* ou DIFTmod) e L_2 (Figura 7.5(b)) o resultado do cálculo dos superpixels para o algoritmo IFTx10.

Se compararmos L_1 e L_2 das Figuras 7.5 (a) e (b) por meio de uma igualdade de rótulos a nível de pixels, a divergência será igual à região vermelha da Figura 7.5(c). Este resultado não corresponde a uma divergência real, dado que do ponto de vista dos superpixels os rótulos não têm uma posição absoluta. Para termos uma medida de divergência mais correta, devemos primeiro encontrar a correspondência de cada superpixel de L_1 na imagem L_2 .

Este procedimento foi realizado da seguinte maneira:

Seja s_i um superpixel de L_1 . O histograma de rótulos dos pixels que pertencem a s_i na imagem L_2 é calculado e o rótulo do superpixel s_i é transformado para o rótulo de maior ocorrência no histograma calculado (gerando assim uma nova imagem de rótulos). Por exemplo, na Figura 7.5(d) o rótulo do superpixel 3 é convertido para o valor 4. Depois de processar todos os superpixels s_i é obtida uma nova imagem de rótulos L_1' (Figura 7.5(e)). A nova comparação entre L_1' e L_2 (Figura 7.5(e-f)) gera um resultado mais exato (Figura 7.5(g)).

Figura 7.5: Correspondência de rótulos de dois mapas de rótulos.

Cálculo da divergência de rótulos Depois de estabelecida a correspondência de rótulos foi calculada a divergência percentual de rótulos nos pixels, por exemplo, para a imagem do nadador na Figura 7.6, foram calculadas as imagens de rótulos para os algoritmos IFTx10 e DIFTmod. As Figuras 7.6(a-b) mostram as bordas de cada superpixel. A divergência de pixels com rótulos distintos é mostrada na Figura 7.6(d), note que a maior divergência ocorre nas bordas de cada superpixel. Essas divergências acontecem em áreas com pouca informação de contraste ou porque são zonas de

empate (Figura 7.6(c)). Por essa razão, foi considerada uma tolerância de 1 pixel nas fronteiras. No final, a Figura 7.6(e) representa as maiores diferenças entre o resultado dos algoritmos IFTx10 e DIFTmod.

Figura 7.6: a-b) Resultados obtidos pela IFTx10 e a DIFTmod respectivamente, com os parâmetros $k = 450$ e $\alpha = 0.06$. c) Imagem original. d) Divergências de rótulos entre os resultados da IFTx10 e a DIFTmod, note que a maioria das diferenças ocorrem em áreas com pouca informação de contraste. e) Divergências de rótulos entre os resultados da IFTx10 e a DIFTmod com tolerância de 1 pixel nas bordas dos superpixels (adjacência 4). Os pixels brancos representam as diferenças nos resultados dos algoritmos.

7.5 Resultados dos experimentos

A seguir são mostrados os resultados de acurácia e o desempenho dos algoritmos DIFT*, DIFTmod e IFTx10, utilizados para o geração de superpixels por meio do algoritmo IFT-SLIC.

7.5.1 Resultados da acurácia

Na Figura 7.7 é representada a porcentagem média de pixels com o rótulo distinto das 50 imagens da base entre os algoritmos DIFT*, DIFTmod e os resultados da IFTx10. Para o algoritmo proposto DIFTmod, a divergência percentual permanece quase constante, apresentando valores bem baixos ($<0,05\%$). Por outro lado, para a DIFT*, a divergência cresce de acordo com o aumento no valor do parâmetro alfa, além de apresentar inconsistências nos mapas de rótulos.

Figura 7.7: Percentagem de divergência entre os resultados dos algoritmos diferenciais (*DIFT**, *DIFTmod*) e a *IFTx10*, para distintos valores dos parâmetros k e α .

A Figura 7.3 apresenta os resultados visuais dos algoritmos *DIFT** e *DIFTmod*. Na Figura 7.3(a) é mostrada a imagem original, na Figura 7.3(b) temos o mapa de rótulos, realçando as áreas inconsistentes (regiões de mesmo rótulo que estão desconexas) (Figura 7.3(c)). Nas Figuras 7.3(d-e) são mostrados os resultados obtidos por meio da *DIFTmod* sem nenhuma inconsistência no mapa de rótulos (Figura 7.3(e)).

Tabela 7.3: Resultados da geração de superpixels pelos algoritmos *DIFT** e *DIFTmod* para os parâmetros $k = 500$ e $\alpha = 0.1$. a) Imagem original de um canguru. b-c) Rótulos obtidos por meio do algoritmo *DIFT** e suas inconsistências (Seção 5.1), observe os pixels vermelhos realçados. d-e) Rótulos obtidos por meio do algoritmo *DIFTmod*. As inconsistências desapareceram com o novo algoritmo.

7.5.2 Resultados do desempenho

A primeira etapa dos três algoritmos corresponde a uma *IFT* sobre todos os pixels da imagem (faixa azul Figura 7.8).

A Figura 7.8 mostra o desempenho de tempo de processamento dos métodos para distintos valores dos parâmetros k e α . Claramente é obtida uma redução do tempo nos fluxos diferenciais em relação ao algoritmo sequencial IFTx10. A relação de economia é mostrada na Figura 7.8(b), em que o desempenho das nove iterações da DIFTmod correspondem a uma pequena fração das nove execuções restantes da IFTx10. Além disso, é destacado em amarelo a faixa de valores α que produz melhores resultados ($\alpha \in [0.04, 0.08]$) tal como recomendado em [ACFM15] e [TMSF17].

Na Figura 7.8 também é possível observar que o desempenho do Algoritmo DIFTmod mantém um desempenho equivalente ao da DIFT* (mas sem gerar inconsistências).

Figura 7.8: Desempenho dos Algoritmos DIFT*, DIFTmod e IFTx10 para distintos valores dos parâmetros k e α .

Claramente, a computação de superpixels através do algoritmo IFT-SLIC utilizando o Algoritmo DIFTmod, obtém baixo custo computacional e resultados com alta acurácia em comparação com o algoritmo IFTx10.

Capítulo 8

Resultados experimentais para a função f_{euc}

Para avaliar o desempenho e a acurácia do algoritmo DIFTmod, foram feitos experimentos com a função f_{euc} (Equação 5.2) para calcular a Transformada de Distância Euclidiana (EDT) em sequências de IFTs. Os experimentos têm a finalidade de comparar os valores do mapa de custo C e os resultados dos mapas de rótulos L , obtidos por meio dos algoritmos diferenciais e os resultados do fluxo sequencial de IFTs [FCC02]. Como resultado foi obtido um desempenho notável e resultados satisfatórios nos mapas C e L para o algoritmo proposto DIFTmod.

8.1 Metodologia

Chamamos de “série” a execução de 50 iterações dos fluxos diferencial (via DIFT* ou DIFTmod) ou sequencial (via IFT’s) sobre a imagem $\hat{I}$, com uma variação no conjunto de sementes em cada uma das iterações. Para o experimento foi criado o seguinte cenário para definir uma “série”:

1. Dados uma imagem $\hat{I}$ de dimensões $m' \times n'$ (n' linhas e m' colunas) e o parâmetro ρ (raio da adjacência circular).
2. Inserir “ d ” sementes no conjunto $\mathcal{S}_1$, sendo “ d ” um número aleatório ($d \in [1, 2, 3, 4]$).
3. Calcular a primeira IFT $_{(\mathcal{S}_1)}$.
4. Para i no intervalo $[2, \dots, 50]$, faça:
 - (a) Escolher um número aleatório “ a ” ($a \in [1, 2, 3, 4]$) e levar em conta as considerações a seguir.
 - No total devem ser feitas “ a ” modificações aleatórias (entre adições e remoções).
 - Se o número de sementes é menor que 5 ($|\mathcal{S}_{i-1}| < 5$), fazer uma adição, isto é para manter um número mínimo de sementes.
 - Na adição realizar um sorteio de um pixel distinto das sementes existentes (semente nova) e adicioná-lo ao conjunto $\Delta_{\mathcal{S}_i}^+$.
 - Na remoção fazer um sorteio aleatório entre as sementes atuais e adicioná-la ao conjunto $\Delta_{\mathcal{S}_i}^-$. Não pode ser selecionada para remoção uma semente que será adicionada na iteração atual.
 - (b) Calcular os mapas C_{IFT} e L_{IFT} por meio do algoritmo IFT com o conjunto de sementes $((\mathcal{S}_{i-1} \cup \Delta_{\mathcal{S}_i}^+) \setminus \Delta_{\mathcal{S}_i}^-)$ como parâmetro.
 - (c) Calcular os mapas C_{DIFT^*} e L_{DIFT^*} por meio do algoritmo DIFT* com os conjunto de sementes $\Delta_{\mathcal{S}_i}^+$ e $\Delta_{\mathcal{S}_i}^-$ como parâmetros.

- (d) Calcular os mapas $C_{DIFTmod}$ e $L_{DIFTmod}$ por meio do algoritmo DIFTmod com os conjunto de sementes $\Delta_{S_i}^+$ e $\Delta_{S_i}^-$ como parâmetros.
- (e) Computar a divergência percentual de valores entre os mapas de custo assim como entre os mapas de rótulos. Sendo a divergência percentual de custo da DIFTmod igual a

$$100 * \frac{\sum_{s \in \mathcal{I}} [C_{IFT}(s) \neq C_{DIFTmod}(s)]}{m' * n'}$$

e a divergência percentual de rótulos igual a

$$100 * \frac{\sum_{s \in \mathcal{I}} [L_{IFT}(s) \neq L_{DIFTmod}(s)]}{m' * n'}$$

Do mesmo modo calcular a divergência entre os mapas da IFT e da DIFT*.

- (f) Atualize o conjunto $\mathcal{S}_i \leftarrow (\mathcal{S}_{i-1} \cup \Delta_{S_i}^+) \setminus \Delta_{S_i}^-$.

8.2 Resultados de uma “série”

Dado que a extensão do caminho $\pi_{r \rightsquigarrow s} \cdot \langle s, t \rangle$ da função de conexidade f_{euc} (Equação 5.2) depende unicamente das posições relativas dos pixels na imagem, os experimentos foram feitos sobre uma imagem sem informação (todos os pixels com o valor 0). A seguir mostramos os resultados de uma “série”, executada sobre uma imagem de dimensões 512x512 com relação de adjacência circular de raio $\rho = 1.5$ (vizinhança 8).

Uma “série” contém 50 execuções para a IFT (execuções sequenciais) e uma (1) execução da IFT mais 49 iterações diferenciais para os algoritmos DIFT* e DIFTmod. Na Figura 8.1 é apresentado a sequência de modificações nos conjuntos Δ_S^+ e Δ_S^- que foram feitas durante uma “série”.

Figura 8.1: Variação de sementes durante as 50 iterações

O desempenho dos algoritmos em cada uma das iterações da “série” é mostrado na Figura 8.2(a), a diferença no esforço computacional é notável para os algoritmos diferenciais. A Figura 8.2(b) apresenta o desempenho acumulativo.

Figura 8.2: (a) *Desempenho durante cada iteração.* (b) *Desempenho acumulativo*

Na Figura 8.3(a) é mostrada a divergência percentual em cada uma das iterações entre os mapas de custo dos algoritmos diferenciais e o mapa de custo da IFT. Note que não existe divergência no mapa de custo da DIFTmod. No entanto, para a DIFT* a divergência é variável ao longo da “série”. A divergência de rótulos é mostrada na Figura 8.3(b), na qual a DIFTmod tem uma divergência menor que 0.03% (originada pelas tie-zones). Na Figura 8.3 as divergências de custo e de rótulos da DIFT* têm um padrão semelhante, onde o custo é diferente, o mais provável é que o rótulo também seja distinto.

Figura 8.3: *Divergência percentual dos algoritmos diferenciais em relação aos resultados obtidos via IFT.* (a) *Divergência de nós com o custo distinto em C .* (b) *Divergência de nós com o rótulo distinto em L .*

Na Tabela 8.1 são ilustradas as divergências da DIFT* nas iterações 15, 16, 17, 18 da “série” apresentada, as regiões vermelhas e azuis representam as divergências nos mapas de custo e de rótulos respectivamente. Na Tabela 8.2 são representados os resultados da DIFTmod para as mesmas iterações, note que não existe divergência nenhuma nessas iterações.

	$i = 15$	$i = 16$	$i = 17$	$i = 18$
Resultado IFT_{s_i}
Mapa C_{DIFT^*}

Tabela 8.1: Divergência dos mapas C e L entre a $DIFT^*$ e a IFT nas iterações 15, 16, 17 e 18.

	$i = 15$	$i = 16$	$i = 17$	$i = 18$
Resultado $IFT_{(s_i)}$
Mapa $C_{DIFT_{mod}}$

Tabela 8.2: Divergência dos mapas C e L entre a $DIFT_{mod}$ e a IFT nas iterações 15, 16, 17 e 18.

Na Figura 8.3(b) foi destacada a região na qual ocorre a maior divergência de rótulos para a DIFTmod (iteração 20). Esta iteração é mostrada na Tabela 8.3. As divergências são muito pequenas ($< 0.03\%$) e ocorrem nos limites das regiões (áreas com zoom na Tabela 8.3(d)), porém não existe divergência no mapa de custo (Tabela 8.3(b)). Para a DIFT* as divergências na mesma iteração são destacadas (Figuras (a) e (c) da Tabela 8.3(a)(c)).

	DIFT* $(\Delta_{S_{20}}^+, \Delta_{S_{20}}^-, c_{19})$	DIFTmod $(\Delta_{S_{20}}^+, \Delta_{S_{20}}^-, c_{19})$	IFT (S_{20})
Mapa de custos
Mapa de rótulos
	(a)	(b)	
	(c)	(d)	

Tabela 8.3: Divergência percentual entre os algoritmos diferenciais e a IFT na iteração 20.

8.3 Resultados para diferentes topologias de grafos

Nas Tabelas 8.4 e 8.5 são apresentadas as médias dos resultados de 30 “séries” para valores de ρ em $\{1, 1.5, 4, 8, 12, 16\}$ (raio da relação de adjacência). O desempenho da IFT ao longo das iterações tem um comportamento constante (Tabela 8.4), já que sempre são modificados todos os nós no grafo. No entanto os algoritmos diferenciais (DIFT* e DIFTmod) têm um melhor desempenho, visto que apenas é atualizado um subconjunto de nós que são alcançados pelas sementes modificadas nos conjuntos Δ_S^+ e Δ_S^- .

Para a DIFT* a divergência nos mapas de custo e de rótulos (Tabela 8.5) persiste ao longo das iterações (em maior ou menor grau), note que a medida que o parâmetro ρ tem seu valor elevado as divergências vão diminuindo, mas ainda existem.

O melhor resultado foi obtido pela DIFTmod, o qual, além de ter um ganho considerável no esforço computacional em relação à IFT (Tabela 8.4), possui uma divergência no mapa de custo nula e uma divergência de rótulos $< 0.03\%$, com variações apenas nas bordas, para todos os valores de ρ (Tabela 8.5).

Tabela 8.4: Resultados da média de 30 “séries” dos três algoritmos para distintos valores de ρ . (a) Média do desempenho por cada iteração. (b) Média acumulativa do desempenho.

Tabela 8.5: Resultados da média de 30 “séries” dos três algoritmos para distintos valores de ρ . (a) Média da divergência no mapa C para cada iteração. (b) Média da divergência no mapa L para cada iteração.

Capítulo 9

Conclusões

O objetivo do presente trabalho foi estender o cálculo da Transformada Imagem-Floresta Diferencial para outros tipos de funções, propósito que foi alcançado preservando os resultados das sequências de iterações da IFT por meio do cálculo diferencial via DIFTmod.

$$\langle IFT_{(\mathcal{S}_1)}, \dots, IFT_{(\mathcal{S}_n)} \rangle = IFT_{(\mathcal{S}_n)} \equiv \langle IFT_{(\mathcal{S}_1)}, DIFTmod_{(\Delta_{\mathcal{S}_2}^+, \Delta_{\mathcal{S}_2}^-, \mathcal{C}_1)}, \dots, DIFTmod_{(\Delta_{\mathcal{S}_n}^+, \Delta_{\mathcal{S}_n}^-, \mathcal{C}_{n-1})} \rangle$$

Os resultados mostraram que nosso algoritmo DIFTmod obtém um desempenho melhor no tempo de execução que resultados obtidos via o fluxo sequencial de IFTs, motivando a pesquisa e aplicação de funções NMI para algum fluxo diferencial, garantindo um resultado com acurácia alta e baixo custo computacional.

9.1 Sugestões para Pesquisas Futuras

As funções de conectividade com aumentos baseados na raiz tratadas neste trabalho englobam um subconjunto de funções NMI. Uma pesquisa futura pode envolver a extensão da DIFTmod para funções de custo com aumentos baseados no prefixo. Assim como explorar aplicações que envolvam sequências de IFTs, que possam ser favorecidas pelo fluxo diferencial da DIFTmod.

Outra pesquisa futura é a possibilidade de criar o algoritmo diferencial da IFT para funções com aumentos negativos $\Delta_C < 0$, por exemplo para a função usada na Transformada Imagem-Floresta Orientada (*Oriented Image Foresting Transform - OIFT*) [MM13b].

Apêndice A

Bandeirantes

Apresentamos um método baseado em grafos para perseguição de bordas em segmentação e rastreamento de curvas, chamado Bandeirantes. Esta abordagem baseia-se em uma função de conectividade com aumentos baseados no prefixo, permitindo incorporar a energia interna dos caminhos em qualquer escala de curvatura e um custo adaptativo da intensidade.

A.1 Motivação

O rastreamento de curvas (*curve tracing*) e a detecção/extração de bordas em imagens, são operações importantes no processamento de imagens e visão computacional, que são problemas fundamentalmente diferentes [Rag04]. A detecção de bordas [DH01] [CLM⁺16] geralmente é o processo de rotular os locais da imagem em que a “taxa de variação” do nível da intensidade é alta, como por exemplo o cálculo do gradiente de uma imagem. A finalidade é extrair todos os pixels que pertencem aos contornos e não rotular uma borda particular. O detector de borda de Canny [Can86] é um algoritmo de extração de bordas popular. Por outro lado, o rastreamento de curvas é um processo de integração de bordas que combina bordas locais (bordas tênues e/ou não contíguas) em curvas relevantes, por exemplo, curvas longas ou contornos fechados para segmentação. Ou seja, queremos identificar uma curva individual e rotulá-la, seguindo-a através da imagem, mesmo que ela acesse outras curvas, desapareça e depois reapareça [Rag04].

Outras aplicações do rastreamento de curvas incluem, rastreamento de objetos, extração de características em imagens médicas (por exemplo, vasos sanguíneos de um angiograma de raios-X [CDPV93]), segmentação de imagens por técnicas de evolução de curvas [KWT87][XP98] e perseguição de borda orientada pelo usuário (*user-steered boundary tracking*) [FUS⁺98][MB98][FCL01][Kan05][MFS11][MFS12].

Em métodos baseados em grafos, a integração das bordas geralmente é baseada em uma função de custo do caminho que combina uma função local indicativa de borda sob a forma de um peso de arco. Essa ideia de transformar o rastreamento de curvas em um problema de encontrar o caminho de custo mínimo em um grafo orientado, foi inicialmente proposta por Martelli, usando o algoritmo heurístico A^* [Mar72][Mar76][Nil80]. Resultados ótimos, de acordo com diferentes critérios, foram posteriormente propostos pelo algoritmo live-wire [FUS⁺98][MB98][FCL01], G-wire [Kan05] e riverbed [MFS11][MFS12].

A.2 Trabalhos relacionados

Karthik Raghupathy aprimorou o trabalho de Carsten Steger [Ste98] para se adequar melhor aos objetivos do rastreamento de curva, emulando a percepção humana usando características globais e locais [Rag04]. As características globais são importantes para lidar com o rastreamento de curvas nas junções e a continuação de curvas fracas que se dissipam ou desaparecem e ressurgem novamente. Em [Rag04], o primeiro passo envolve classificar pixels como pontos de curva ou não, dependendo se eles se encontram em alguma curva na imagem (detecção de borda). As curvas são então construídas a partir de um ponto inicial (selecionado pelo usuário), conectando um ponto de curva apropriado na vizinhança da curva atual. Como desvantagens, este algoritmo exige uma seleção apropriada de oito parâmetros e uma binarização dos pontos de curva o qual é um passo crítico sujeito à falha, além disso não é possível especificar um ponto de destino da curva, como no live-wire, já que as conexões terminam quando não há mais pontos de curva na vizinhança do pixel atual.

Contorno ativo

O modelo de contorno ativo [KWT87][XP98], também chamado de *snakes*, é baseado na minimização de energia, em que a curva *spline* deformável é influenciada pelas forças de restrição da imagem que a puxam para contornos de objetos e forças internas (como curvatura) que resistem à deformação. O modelo deforma uma curva inicial (especificada pelo usuário) em direção a borda alvo para delinear o contorno do objeto em uma imagem 2D possivelmente com ruído. A forma final da fronteira é difícil de se prever ou controlar, então em caso de falha, todo o processo deve ser repetido com uma nova curva inicial usando parâmetros diferentes. Como apontado por H. W. Kang [Kan05], no contexto da segmentação interativa, o live-wire fornece um controle maior para os usuários, já que o caminho desejado pode ser selecionado de forma interativa a partir de vários caminhos candidatos. O algoritmo do snakes foi posteriormente combinado com métodos baseados em grafos, levando ao modelo de contorno ativo geodésico (*geodesically linked active contour model*), que basicamente consiste em um conjunto de vértices conectados por caminhos de custo mínimo [MC09][MBC15]. O resultado é uma curva fechada (suave em algumas partes), sobre a qual uma função de energia de borda ou região pode ser calculada.

Dado que a abordagem proposta é baseada em grafos, ela poderia ser utilizada nesse modelo, mas note que neste trabalho não estamos limitando o escopo para contornos fechados.

Métodos baseados em grafos

Os métodos baseados em grafos live-wire [FUS⁺98][MB98][FCL01], G-wire [Kan05] e riverbed [MFS11][MFS12] calculam caminhos para todos os pixels a partir de pixels iniciais (sementes), também conhecidos como pontos de ancoragem. Em uma segmentação de imagens orientada pelo usuário, o caminho calculado do ponto de ancoragem anterior até a posição atual do mouse é mostrado ao usuário, à medida que ele move o cursor, de modo que o usuário possa selecionar o caminho desejado de forma interativa. Após selecionar um caminho, o processo é repetido iniciando um novo caminho com ponto inicial no último ponto de ancoragem selecionado. Todos os caminhos selecionados anteriores são mantidos inalterados durante o algoritmo, de modo que seus nós não possam ser revisitados.

Live-wire e riverbed são baseados no arcabouço da IFT [FSL04] usando diferentes funções de custo do caminho. O algoritmo riverbed pode reduzir significativamente o número de interações do

usuário (pontos de ancoragem) em relação ao live-wire para objetos com formas complexas e bom contraste, mas o oposto é verdadeiro para objetos com baixo contraste e forma mais regular, já que o live-wire também pode levar em consideração a distância espacial (Figura A.1), que corresponde ao termo de primeira ordem da energia interna do snakes. G-wire incorpora o termo de segunda ordem (termo de curvatura) da energia interna com base em programação dinâmica de alta ordem envolvendo uma formulação em um grafo multidimensional, preservando assim a otimalidade dos caminhos calculados. No entanto, é computacionalmente caro, exigindo um grafo tridimensional para resolver um problema em uma imagem 2D. Além disso, seu termo de curvatura é definido em uma escala muito local, pois analisa apenas pixels imediatamente vizinhos. Como consequência, o G-wire pode melhorar os resultados da segmentação em imagens ruidosas, mas não pode resolver adequadamente curvas maiores, como demonstrado nos experimentos. Note que, na prática, a curvatura multi-escala em G-wire é proibitiva, já que no caso geral com uma função de conexidade que se estende por m arcos, a dimensionalidade do grafo seria $2 + (m - 1)$ [Kan05].

Figura A.1: Métodos baseados em grafos. (a) Live-wire. (b) Riverbed.

Semelhante ao live-wire e o riverbed, o bandeirantes também é baseado no arcabouço da IFT [FSL04], conservando sua baixa complexidade computacional de $O(N \cdot \log N)$ para um grafo esparsos com N nós.

A.2.1 Funções de custo

Live-wire

A função do live-wire é dada por:

$$\begin{aligned} f_{LW}(\langle t \rangle) &= \begin{cases} 0 & \text{se } t = s^* \\ +\infty & \text{caso contrário} \end{cases} \\ f_{LW}(\pi_s \cdot \langle s, t \rangle) &= f_{LW}(\pi_s) + c(s, t) \end{aligned} \quad (\text{A.1})$$

na qual s^* é o pixel semente (ponto de partida). Geralmente temos $c(s, t) = \gamma \cdot c_{ext}(s, t) + \varphi \cdot c_{in}(s, t)$, em que c_{ext} e c_{in} denotam, respectivamente, a energia externa e interna.

A energia externa geralmente está relacionada à magnitude do gradiente, tal como seu complemento $c_{ext}(s, t) = K - \frac{\|\nabla I(s)\| + \|\nabla I(t)\|}{2}$, e a energia interna do live-wire é um termo de primeira ordem $c_{in}(s, t) = \|s - t\|$. Observe que f_{LW} é uma função de custo do caminho monotonicamente incremental para $c(s, t) \geq 0$.

Riverbed

A regra de extensão do caminho do riverbed é dada por: $f_{RB}(\pi_s \cdot \langle s, t \rangle) = c(s, t)$. A função f_{RB} não é monotonicamente incremental e sua otimalidade pela IFT é assegurada por outro critério de otimalidade [MFS12].

G-wire

A regra de extensão do G-wire é dada por:

$$f_{GW}(\pi = \langle t_1, t_2, \dots, t_n \rangle) = f_{GW}(\langle t_1, \dots, t_{n-1} \rangle) + c(t_{n-2}, t_{n-1}, t_n) \quad (\text{A.2})$$

em que geralmente temos $c(t_{n-2}, t_{n-1}, t_n) = \gamma \cdot c_{ext}(t_{n-1}, t_n) + c_{in}(t_{n-2}, t_{n-1}, t_n)$ e sua energia interna também inclui o termo de segunda ordem (termo de curvatura) $c_{in}(t_{n-2}, t_{n-1}, t_n) = \varphi \cdot \|t_n - t_{n-1}\|^2 + \beta \cdot \|t_n - 2 \cdot t_{n-1} + t_{n-2}\|^2$.

Observe que f_{GW} não é uma função monotonicamente incremental, posto que seu incremento de custo depende de t_{n-2} .

A.3 Bandeirantes

A principal motivação do novo método proposto é superar a limitação do G-wire, que o termo de curvatura na G-wire é definido em uma escala muito local, analisando apenas pixels imediatamente vizinhos. A curvatura multi-escala no G-wire é proibitiva, já que no caso geral com uma função de custo que se estende por m arcos, a dimensionalidade do grafo seria $2 + (m - 1)$ para o cálculo de curvas em uma imagem 2D [Kan05].

A ideia do Bandeirantes é usar funções de custo de caminho que satisfaçam a equação:

$$\begin{aligned} f_B(\langle t \rangle) &= f_{LW}(\langle t \rangle) \\ f_B(\pi = \langle t_1, t_2, \dots, t_n \rangle) &= f_B(\langle t_1, \dots, t_{n-1} \rangle) + c(t_1, \dots, t_n) \end{aligned} \quad (\text{A.3})$$

na qual $c(t_1, \dots, t_n) \geq 0$ é uma função de várias variáveis.

Esta função f_B não é MI, portanto, quebra o princípio de otimalidade no algoritmo da IFT.

Apesar de f_B não ser MI, o algoritmo da IFT com f_B é ótimo do ponto de vista a seguir. Cada arco $\langle s, t \rangle \in \mathcal{A}$ é processado uma única vez durante o Algoritmo 1 depois que s é removido de $\mathcal{Q}$ na Linha 8. Considere um grafo direcionado ponderado $\mathbf{G}' = \langle \mathcal{I}, \mathcal{A} \rangle$, com pesos criados dinamicamente durante a execução da IFT com f_B em G , de modo que os pesos de arco $\omega(\langle s, t \rangle)$ para $\langle s, t \rangle \in \mathcal{A}$ são definidos dentro do loop, logo após a Linha 10, como $w(s, t) \stackrel{\text{def}}{=} c(t_1, \dots, t_n = t)$, para $\pi_s^P = \langle t_1, \dots, t_{n-1} = s \rangle$.

A floresta P calculada pela IFT com f_B em $\mathbf{G}$ é uma floresta de caminho ótimos para f_{LW} em $\mathbf{G}'$, usando $c(s, t) = \omega(\langle s, t \rangle)$ na Equação A.1. Assim, o resultado da IFT com f_B é um resultado ótimo para o live-wire sobre um grafo criado dinamicamente. O nome *Bandeirantes*¹, provém dessa

¹Os Bandeirantes foram colonos portugueses do século XVII no Brasil e caçadores de fortunas da região São Paulo. Eles lideraram expedições chamadas bandeiras que penetraram no interior do Brasil, ao sul e a oeste da

observação, em alusão a exploradores que analisam um território desconhecido que apenas pode ser conhecido em tempo real, semelhante aos pesos de arco que são calculados em tempo de execução. Outras propriedades teóricas do Bandeirantes são discutidas no artigo [CMM17].

A.4 Resultados experimentais e conclusões

A Figura A.2 mostra o resultado do Bandeirantes como uma abordagem de perseguição de bordas em comparação com outros métodos. Note que o G-wire produz um resultado semelhante em relação ao live-wire, mas com um contorno um pouco mais suave. Para o Bandeirantes, foi usado $c(t_1, \dots, t_n) = \|t_n - 2 \cdot t_{n-k} + t_{n-2k}\|^p + \varphi^p \cdot (W(t_n) + W(t_{n-1}))^p$ na Equação A.3, com os valores $p = 2$, $\varphi = 0.01$, $k = 15$ e $W(t)$ denota o complemento da magnitude do gradiente.

Bandeirantes dá o contorno mais regular, seguindo mais de perto a dura-máter, devido à sua análise de curvatura envolvendo até o 30º antecessor no caminho. O segundo exemplo na Figura A.3 mostra que o G-wire tem dificuldade em rastrear contornos ruidosos com oscilações em escala local, dando preferência para uma curva suave na escala local, mesmo que ela tenha um alto grau de curvatura de longo prazo (Figura A.3(a)). O método do Bandeirantes pode perseguir essas curvas com perturbações locais, quando elas apresentam uma baixa curvatura de longo prazo (Figura A.3(b)).

Figura A.2: O rastreamento do contorno externo do cérebro em uma imagem MR-T1: (a) riverbed, (b) live-wire, (c) G-wire e (d) Bandeirantes.

Figura A.3: Perseguição de bordas sobre uma imagem sintética: (a) G-wire, e (b) Bandeirantes.

Foram realizados experimentos quantitativos para avaliar o desempenho do Bandeirantes, em comparação com os métodos live-wire, riverbed e G-wire, em uma tarefa de rastreamento de curvas em um sistema de avaliação automática de provas. Também foram testados “Snake with Gradient Vector Flow” [XP98] da Active Contour Library em C/C++², mas seus resultados se mostraram sensíveis à inicialização e muito lentos.

linha Tordesillas de 1494. Ao se aventurarem em regiões não mapeadas em busca de lucro e aventura, expandiram as fronteiras efetivas da colônia brasileira.

²<https://github.com/HiDiYANG/ActiveContour>

Em sistemas de avaliação automática, como questionários de múltipla escolha (*Auto Multiple Choice* - AMC³), geralmente têm questionários nos quais os alunos preenchem completamente as caixas relevantes. Após o exame, as folhas de respostas com as caixas marcadas são digitalizadas e podem ser detectadas automaticamente pelo AMC, usando Reconhecimento de Marcas Ópticas (*Optical Mark Recognition* - OMR). No experimento, para avaliar a capacidade de programação dos alunos, eles foram convidados a vincular os espaços vazios em um determinado código parcial às opções corretas de uma grande lista de comandos, conectando seus pontos traçando curvas suaves (Figura A.4(a)).

Figura A.4: (a) Exemplo de uma folha de resposta preenchida e digitalizada. Os caminhos calculados por Bandeirantes para todas as opções a partir da: (b) primeira lacuna e da (c) segunda lacuna. (d) Os melhores pares correspondentes calculados pelo algoritmo húngaro.

Para cada método avaliado, primeiro foram calculados os caminhos a partir dos pontos $\{s_1, \dots, s_M\}$ (espaços vazios) para todos os pontos $\{d_1, \dots, d_N\}$ na lista de comandos (Figuras A.4(b-c)), o problema de atribuição é resolvido pelo algoritmo Húngaro⁴ na matriz C (Equação A.4), que encontra o melhor par correspondente (Figura A.4(d)). Observe que o método de rastreamento de curvas em [Rag04] não pôde ser usado nesta avaliação, porque não permite a especificação dos pixels de destino.

$$C_{M,N} = \begin{pmatrix} f(\pi_{s_1 \rightsquigarrow d_1}) & f(\pi_{s_1 \rightsquigarrow d_2}) & \cdots & f(\pi_{s_1 \rightsquigarrow d_N}) \\ f(\pi_{s_2 \rightsquigarrow d_1}) & f(\pi_{s_2 \rightsquigarrow d_2}) & \cdots & f(\pi_{s_2 \rightsquigarrow d_N}) \\ \vdots & \vdots & \ddots & \vdots \\ f(\pi_{s_M \rightsquigarrow d_1}) & f(\pi_{s_M \rightsquigarrow d_2}) & \cdots & f(\pi_{s_M \rightsquigarrow d_N}) \end{pmatrix} \quad (\text{A.4})$$

No caso do live wire ($f = f_{LW}$), a melhor configuração encontrada foi obtida usando $c(s, t) = \varphi^p \cdot (I(s) + I(t))^p + \|s - t\|$ na Equação A.1, na qual $p = 9$, $\varphi = 0.03$ e $I(t)$ é a intensidade da imagem. Para riverbed ($f = f_{RB}$), a melhor configuração foi $c(s, t) = I(s) + I(t)$. Para G-wire ($f = f_{GW}$), os melhores resultados foram obtidos usando $c(t_{n-2}, t_{n-1}, t_n) = \gamma^p \cdot (I(t_n) + I(t_{n-1}))^p + \varphi \cdot \|t_n - t_{n-1}\| + \|t_n - 2 \cdot t_{n-1} + t_{n-2}\|^p$ na Equação A.2, com os valores $p = 9$, $\gamma = 0.03$ e $\varphi = 0.01$. Para Bandeirantes ($f = f_B$), foi utilizada $c(t_1, \dots, t_n) = \|t_n - 2 \cdot t_{n-k} + t_{n-2k}\|^p + \varphi^p \cdot (|I(t_n) - \frac{\sum_{i=n-2k}^{n-k} I(i)}{k+1}| + I(t_n) + I(t_{n-1}))^p$ na Equação A.3, em que $p = 9$, $\varphi = 0.03$ e $k = 16$.

A componente $|I(t_n) - \frac{\sum_{i=n-2k}^{n-k} I(i)}{k+1}|$ em f_B ajuda o Bandeirantes a seguir curvas fracamente desenhadas. Empates de custo foram raramente detectados (menos de 0,01%) no Bandeirantes, quando foram usados custos em formato de ponto flutuante de dupla precisão.

Cada método é calculado usando sua própria função de custo do caminho. O algoritmo IFT é

³<http://home.gna.org/auto-qcm/>

⁴<https://github.com/saebyn/munkres-cpp>

usado para computar o live-wire, riverbed e bandeirantes, enquanto o G-wire é calculado usando uma programação dinâmica de alta ordem em um grafo tridimensional. Em relação ao preenchimento das entradas da matriz C , foram consideradas duas versões para cada método (riverbed/live-wire/G-wire/bandeirantes), uma usando seus próprios custos ($f_{RB}/f_{LW}/f_{GW}/f_B$) e outra considerando em C os custos atribuídos por f_B dos caminhos calculados.

A Tabela A.1 mostra os resultados experimentais, na qual os números indicam a taxa de atribuição média correta das curvas por imagem. A superioridade do método proposto é evidente a partir dos experimentos. Além disso, o Bandeirantes foi 81% mais rápido que o G-wire. Os erros ocorrem principalmente em situações desafiadoras incluídas no conjunto de dados, como curvas que passam sobre elas mesmas gerando um laço (que não pode ser armazenado em um mapa de predecessores), curvas que passam muito perto de outra opção de destino e curvas desenhadas com alta curvatura. É importante notar que na prática, os alunos são instruídos a desenhar apenas curvas simples com baixa curvatura, de modo que a taxa de erro seria menor.

	Seu próprio custo em C	f_B em C
Riverbed	14.19%	49.88%
Live wire	26.33%	57.60%
G-wire	36.50%	57.03%
Bandeirantes	95.24%	95.24%

Tabela A.1: A taxa média de atribuições corretas das curvas por imagem para diferentes métodos.

O banco de dados composto por 70 imagens digitalizadas e o código-fonte em C/C++ estão disponíveis⁵.

⁵<http://www.vision.ime.usp.br/~mtejadac/bandeirantes.html>

Referências Bibliográficas

- [ACFM15] Eduardo B. Alexandre, Ananda S. Chowdhury, Alexandre X. Falcão e Paulo A. V. Miranda. IFT-SLIC: A general framework for superpixel generation based on simple linear iterative clustering and image foresting transform. Em *2015 28th SIBGRAP Conference on Graphics, Patterns and Images*, páginas 337–344. IEEE, 2015. [2](#), [32](#), [51](#), [53](#), [56](#), [61](#)
- [ASS⁺12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua e Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012. [51](#), [53](#)
- [Ber04] Felipe P. G. Bergo. Segmentação interativa de volumes baseada em regiões. Dissertação de Mestrado, Instituto de Computação, Universidade de Campinas, Brasil, 2004. [viii](#), [22](#), [23](#), [24](#), [25](#), [26](#), [27](#)
- [Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986. [73](#)
- [CCFM17] Marcos A. T. Condori, Fábio A. M. Cappabianco, Alexandre X. Falcão e Paulo A. V. Miranda. Extending the differential image foresting transform to root-based path-cost functions with application to superpixel segmentation. Em *2017 30th SIBGRAP Conference on Graphics, Patterns and Images (SIBGRAP)*, páginas 7–14, Oct 2017. [3](#)
- [CDPV93] C. Coppini, M. Demi, R. Poli e G. Valli. An artificial vision system for x-ray images of human coronary arteries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):156–162, Feb 1993. [73](#)
- [CFYU12] Fábio A. M. Cappabianco, Alexandre X. Falcão, Clarissa L. Yasuda e Jayaram K. Udupa. Brain tissue MR-image segmentation via optimum-path forest clustering. *Computer Vision and Image Underst.*, 116(10):1047–1059, 2012. [11](#)
- [CLM⁺16] Fábio A. Cappabianco, Lucas S. Lellis, Paulo A. V. Miranda, Jaime S. Ide e Lilianne Mujica-Parodi. Edge detection robust to intensity inhomogeneity: A 7T MRI case study. Em *XXI IberoAmerican Congress on Pattern Recognition (CIARP)*, Lima, Peru, Nov 2016. accepted, to appear. [73](#)
- [CMM17] Marcos A. T. Condori, Lucy A. C. Mansilla e Paulo A. V. Miranda. Bandeirantes: A graph-based approach for curve tracing and boundary tracking. Em *Mathematical Morphology and Its Applications to Signal and Image Processing*, páginas 95–106, May 2017. [2](#), [3](#), [11](#), [77](#)
- [DH01] David Donoho e Xiaoming Huo. Applications of beamlets to detection and extraction of lines, curves and objects in very noisy images. Em *In Proceedings of NSIP*, 2001. [73](#)

- [Dia69] Robert B. Dial. Algorithm 360: Shortest-path forest with topological ordering [h]. *Commun. ACM*, 12(11):632–633, Novembro 1969. 18
- [Dij59] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959. 11
- [Fal09] Alexandre X. Falcão. *Anotações de aula - mo815: Processamento de imagens usando grafos*. <http://www.ic.unicamp.br/~afalcao/mo815-grafos/index.html>, 2009. 2
- [FB04] Alexandre X. Falcão e Felipe P. G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Transactions on Medical Imaging*, 23(9):1100–1108, 2004. 1, 21, 24
- [FCC02] Alexandre X. Falcão, Luciano F. Costa e Bruno S. Cunha. Multiscale skeletons by image foresting transform and its application to neuromorphometry. *Pattern Recognition*, 35(7):1571 – 1582, 2002. 2, 63
- [FCL01] Alexandre X. Falcão, Bruno S. Cunha e Roberto A. Lotufo. Design of connected operators using the image foresting transform, 2001. 1, 73, 74
- [FSL04] Alexandre X. Falcão, Jorge Stolfi e Roberto A. Lotufo. The image foresting transform: Theory, algorithms, and applications. *IEEE TPAMI*, 26(1):19–29, 2004. 1, 2, 11, 13, 16, 74, 75
- [FUM00] Alexandre X. Falcão, Jayaram K. Udupa e Flavio Miyazawa. An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly. *IEEE Transactions on Medical Imaging*, 19(1):55–62, Jan 2000. 1, 2, 11
- [FUS⁺98] Alexandre X. Falcão, Jayaram K. Udupa, Supun Samarasekera, Shoba Sharma, Bruce E. Hirsch e Roberto A. Lotufo. User-steered image segmentation paradigms: Live-wire and live-lane. *Graph. Models and Image Proc.*, 60:233–260, 1998. 9, 73, 74
- [HC01] Gabor T. Herman e Bruno M. Carvalho. Multiseeded segmentation using fuzzy connectedness. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(5):460–474, Maio 2001. 2
- [Kan05] Hyung W. Kang. G-wire: A livewire segmentation algorithm based on a generalized graph formulation. *Pattern Recognition Letters*, 26(13):2042–2051, Oct 2005. 2, 73, 74, 75, 76
- [KWT87] Michael Kass, Andrew Witkin e Demetri Terzopoulos. Snakes: Active contour models. *Intl. Journal of Computer Vision*, 1:321–331, 1987. 73, 74
- [LFZ02] Roberto de A. Lotufo, Alexandre X. Falcão e Francisco A. Zampiroli. IFT-watershed from gray-scale marker. Em *Computer Graphics and Image Processing, 2002. Proceedings. XV Brazilian Symposium on*, páginas 146–152. IEEE, 2002. 11
- [LM17] Leissi M. C. Leon e Paulo A. V. Miranda. Multi-object segmentation by hierarchical layered oriented image foresting transform. Em *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, páginas 79–86, Oct 2017. 2
- [Man14] Lucy A. C. Mansilla. Transformada imagem-floresta com funções de conexidade não suaves: pesos adaptativos, polaridade de borda e restrições de forma. Dissertação de Mestrado, Instituto de Matemática e Estatística, University of São Paulo, Brasil, 2014. 1, 17
- [Mar72] Alberto Martelli. Edge detection using heuristic search methods. *Computer Graphics and Image Processing*, 1:169–182, 1972. 73

- [Mar76] Alberto Martelli. An application of heuristic search methods to edge and contour detection. *Communications of the ACM*, 19(2):73–83, 1976. [73](#)
- [MB98] Eric Mortensen e William Barrett. Interactive segmentation with intelligent scissors. *Graph. Models and Image Proc.*, 60:349–384, 1998. [73](#), [74](#)
- [MBC15] Julien Mille, Sébastien Boudleux e Laurent D. Cohen. Combination of piecewise-geodesic paths for interactive segmentation. *Intl. Journal of Computer Vision*, 112(1):1–22, 2015. [74](#)
- [MC09] Julien Mille e Laurent D. Cohen. Geodesically linked active contours: Evolution strategy based on minimal paths. Em *Proc. of the Second Intl. Conference on Scale Space and Variational Methods in Computer Vision*, páginas 163–174, Berlin, Heidelberg, 2009. [74](#)
- [MFaRB08] Paulo A. V. Miranda, Alexandre X. Falcão, Anderson Rocha e Felipe P. G. Bergo. Object delineation by κ-connected components. *EURASIP J. Adv. Signal Process.*, 2008:159:1–159:14, Janeiro 2008. [2](#)
- [MFS11] Paulo A. V. Miranda, Alexandre X. Falcão e Thiago V. Spina. The riverbed approach for user-steered image segmentation. Em *Image Processing (ICIP), 2011 18th IEEE International Conference on*, páginas 3133–3136, Sept 2011. [2](#), [73](#), [74](#)
- [MFS12] Paulo A.V. Miranda, Alexandre X. Falcão e Thiago V. Spina. Riverbed: A novel user-steered image segmentation method based on optimum boundary tracking. *Image Processing, IEEE Transactions on*, 21(6):3042–3052, June 2012. [2](#), [73](#), [74](#), [76](#)
- [MM13a] Lucy A. C. Mansilla e Paulo A. V. Miranda. *Image Segmentation by Oriented Image Foresting Transform with Geodesic Star Convexity*, páginas 572–579. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. [2](#)
- [MM13b] Lucy A. C. Mansilla e Paulo A.V. Miranda. Image segmentation by oriented image foresting transform: Handling ties and colored images. Em *18th Intl. Conf. on Digital Signal Processing*, páginas 1–6, Greece, Jul 2013. [2](#), [71](#)
- [MM14] Paulo A. V. Miranda e Lucy A. C. Mansilla. Oriented image foresting transform segmentation by seed competition. *IEEE Transactions on Image Processing*, 23(1):389–398, Jan 2014. [1](#), [2](#), [11](#), [13](#)
- [MMC13] Lucy A. C. Mansilla, Paulo A. V. Miranda e Fábio A. M. Cappabianco. Image segmentation by image foresting transform with non-smooth connectivity functions. Em *2013 XXVI Conference on Graphics, Patterns and Images*, páginas 147–154. IEEE, 2013. [2](#)
- [MMC16] Lucy A. C. Mansilla, Paulo A.V. Miranda e Fábio A. M. Cappabianco. Oriented image foresting transform segmentation with connectivity constraints. Em *Image Processing (ICIP), 2016 IEEE International Conference on*, páginas 2554–2558. IEEE, 2016. [2](#), [9](#)
- [Moy15] Nikolas Moya. Segmentação interativa de múltiplos objetos 3d em imagens médicas por cortes Ótimos em grafo. Dissertação de Mestrado, Instituto de Computação, Universidade de Campinas, Brasil, Mar 2015. [39](#)
- [Nil80] Nils J. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1980. [73](#)
- [PFMM07] João P. Papa, Alexandre X. Falcão, Celso T. N. Miranda, Paulo A. V. and Suzuki e Nelson D. A. Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. *Comput. Vis. Image Underst.*, páginas 337–348, 2007. [1](#)

- [Rag04] Karthik Raghupathy. *Curve tracing and curve detection in images*. Cornell University, August, Aug 2004. A Thesis Presented to the Faculty of the Graduate School of Cornell University. [73](#), [74](#), [78](#)
- [RFSR13] Paulo E. Rauber, Alexandre X. Falcão, Thiago V. Spina e Pedro J. Rezende. Interactive segmentation by image foresting transform on superpixel graphs. Em *2013 XXVI Conference on Graphics, Patterns and Images*, páginas 131–138, Aug 2013. [51](#)
- [RKB04] Carsten Rother, Vladimir Kolmogorov e Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Agosto 2004. [9](#), [57](#)
- [SCMS13] Robin Strand, Krzysztof C. Ciesielski, Filip Malmberg e Punam K. Saha. The minimum barrier distance. *Comput. Vis. Image Underst.*, 117(4):429–437, Abril 2013. [2](#)
- [SDS13] Guang Shu, Afshin Dehghan e Mubarak Shah. Improving an object detector and extracting regions using superpixels. Em *2013 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3721–3727, June 2013. [51](#)
- [Ste98] Carsten Steger. An unbiased detector of curvilinear structures. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(2):113–125, Fevereiro 1998. [74](#)
- [TC17] Lucas Teixeira e Margarita Chli. Real-time local 3d reconstruction for aerial inspection using superpixel expansion. Em *2017 IEEE International Conference on Robotics and Automation (ICRA)*, páginas 4560–4567, May 2017. [51](#)
- [TFC04] Ricardo S. Torres, Alexandre X. Falcão e Luciano F. Costa. A graph-based approach for multiscale shape analysis. *Pattern Recognition*, 37(6):1163 – 1174, 2004. [2](#)
- [TMSF17] Anderson M. Tavares, Paulo A. V. Miranda, Thiago V. Spina e Alexandre X. Falcão. A supervoxel-based solution to resume segmentation for interactive correction by differential image-foresting transforms. Em *Mathematical Morphology and Its Applications to Signal and Image Processing*, páginas 107–118, May 2017. [61](#)
- [WLG⁺17] Murong Wang, Xiabi Liu, Yixuan Gao, Xiao Ma e Nouman Q. Soomro. Superpixel segmentation: A benchmark. *Signal Processing: Image Communication*, 56(Supplement C):28 – 39, 2017. [51](#)
- [WLYY11] Shu Wang, Huchuan Lu, Fan Yang e M. H. Yang. Superpixel tracking. Em *2011 International Conference on Computer Vision*, páginas 1323–1330, Nov 2011. [51](#)
- [XP98] Chenyang Xu e J. L. Prince. Snakes, shapes, and gradient vector flow. *Trans. Img. Proc.*, 7(3):359–369, Março 1998. [73](#), [74](#), [77](#)