Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação

Utilizando Mapas de Conectividade Fuzzy no Desenvolvimento de Algoritmos Reparadores de Imagens Binárias 3D

Íria Caline Saraiva Cosme

Natal/RN
Agosto de 2008

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação

Utilizando Mapas de Conectividade Fuzzy no Desenvolvimento de Algoritmos Reparadores de Imagens Binárias 3D

Exame de dissertação submetido ao Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como parte dos requisitos para a obtenção do grau de Mestre em Sistemas e Computação.

Íria Caline Saraiva Cosme

Natal/RN
Agosto de 2008

Utilizando Mapas de Conectividade Fuzzy no Desenvolvimento de Algoritmos Reparadores de Imagens Binárias 3D

Íria Caline Saraiva Cosme

Este exame de dissertação foi avaliado e considerado aprovado pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Prof. Dr. Bruno Motta de Carvalho
Orientador

Prof. Dr. Marcelo Ferreira Siqueira
Orientador

Profa. Dra. Thaís Vasconcelos Batista
Coordenador do Programa

Banca Examinadora:

Prof. Dr. Bruno Motta de Carvalho
Presidente

Prof. Dr. Marcelo Ferreira Siqueira

Prof. Dr. Benjamín René Callejas Bedregal

Prof. Dr. Luiz Marcos Garcia Gonçalves

"Você nunca sabe que resultados virão da sua ação. Mas se você não fizer nada, não existirão resultados."

Mahatma Gandhi.

Resumo

Uma imagem binária 3D é considerada bem-composta se, e somente se, a união das faces compartilhadas pelos *voxels* do *foreground* e do *background* da referida imagem é uma superfície em $\mathbb{R}^3$. Imagens bem-compostas se beneficiam de propriedades topológicas desejáveis, as quais nos permitem simplificar e otimizar algoritmos amplamente usados na computação gráfica, visão computacional e processamento de imagens. Estas vantagens têm motivado o desenvolvimento de algoritmos para reparar imagens bi e tridimensionais que não sejam bem-compostas. Estes algoritmos são conhecidos como algoritmos reparadores. Nesta dissertação, propomos dois algoritmos reparadores, um aleatório e um determinístico. Ambos são capazes de fazer reparos topológicos em imagens binárias 3D, produzindo imagens bem-compostas similares às imagens originais. A idéia fundamental por trás de ambos algoritmos é mudar iterativamente a cor atribuída de alguns pontos da imagem de entrada de 0 (*background*) para 1 (*foreground*) até a imagem se tornar bem-composta. Os pontos cujas cores são mudadas pelos algoritmos são escolhidos de acordo com seus valores no mapa de conectividade *fuzzy*, resultante do processo de segmentação da imagem. O uso do mapa de conectividade *fuzzy* garante que um subconjunto dos pontos escolhidos pelo algoritmo em qualquer iteração seja um com a menor afinidade com o *background* dentre todas as escolhas possíveis.

Área de Concentração: Teoria e Inteligência Computacional
Palavras-chave: Imagens bem-compostas, segmentação *fuzzy*, algoritmos reparadores.

Abstract

A 3D binary image is considered well-composed if, and only if, the union of the faces shared by the foreground and background voxels of the image is a surface in $\mathbb{R}^3$. Well-composed images have some desirable topological properties, which allow us to simplify and optimize algorithms that are widely used in computer graphics, computer vision and image processing. These advantages have fostered the development of algorithms to repair bi-dimensional (2D) and three-dimensional (3D) images that are not well-composed. These algorithms are known as repairing algorithms. In this dissertation, we propose two repairing algorithms, one randomized and one deterministic. Both algorithms are capable of making topological repairs in 3D binary images, producing well-composed images similar to the original images. The key idea behind both algorithms is to iteratively change the assigned color of some points in the input image from 0 (background) to 1 (foreground) until the image becomes well-composed. The points whose colors are changed by the algorithms are chosen according to their values in the fuzzy connectivity map resulting from the image segmentation process. The use of the fuzzy connectivity map ensures that a subset of points chosen by the algorithm at any given iteration is the one with the least affinity with the background among all possible choices.

Area of Concentration: Theory and Computational Intelligence
Key words: Well-composed images, fuzzy segmentation, repair algorithms.

Sumário

Lista de Símbolos e Abreviaturas **i**

1 Introdução **1**

1.1 Contribuições 4

1.2 Organização do trabalho 5

2 Trabalhos Relacionados **6**

3 Imagens Digitais Binárias 3D Bem-Compostas **11**

3.1 Definições Básicas 12

3.2 Imagens Digitais 3D Bem-compostas 14

4 Segmentação Fuzzy de Imagens Digitais **17**

4.1 Definições Básicas 18

4.2 Segmentação de Imagens 19

4.2.1 Segmentação Orientada a Regiões 21

4.3 Segmentação *Fuzzy* de Imagens 22

4.3.1 Segmentação *Fuzzy* Simultânea de Múltiplos Objetos 23

5 Reparação de Imagens Bem-Compostas Utilizando Mapas de Conectividade *Fuzzy* **31**

5.1 Ran3DWCFS - Um Algoritmo Reparador Aleatório para Imagens Bem-Compostas utilizando Segmentação *Fuzzy* 32

5.1.1 Eliminação das Configurações Críticas 33

5.2 Det3DWCFS - Um Algoritmo Reparador Determinístico para Imagens Bem-Compostas utilizando Segmentação *Fuzzy* 38

5.2.1 Eliminação das Configurações Críticas 39

5.2.2 Uma simplificação do Det3DWCFS 40

5.3 Experimentos 40

5.4 Análise Comparativa dos Resultados 45

5.4.1 Análise em Relação ao Valor dos Desvios Padrão 47

5.4.2 Análise em Relação à Quantidade de Pontos Modificados 47

5.4.3 Análise em Relação à Divisão do Total dos Graus de Pertinência pela Quantidade de Pontos Modificados 48

5.4.4 Análise em Relação ao Nível de Ruído das Imagens 49

6 Conclusão **51**

6.1 Trabalhos Futuros 52

Lista de Figuras

2.1 Configuração crítica de uma imagem 2D não bem-composta. Os quadrados pretos representam pontos do *foreground* e os brancos, pontos do *background*. 6

2.2 Alça feita a partir de um volume da substância branca do cérebro humano e o resultado da aplicação da técnica do corte desta alça e do preenchimento do ofício. 8

3.1 Um conjunto desconectado pode ter uma digitalização conectada. Figura extraída de Latecki, Eckhardt e Rosenfeld [LER95]. 11

3.2 Análogo contínuo do conjunto $X = \{a, b, c, d\} \subset \mathbb{Z}^3$. Figura extraída de Siqueira et al. [SLT$^+$08]. 13

3.3 Tipos de configurações críticas: (a) Configuração Crítica 1. (b) Configuração Crítica 2. Para um melhor entendimento, ambas as imagens mostram apenas voxels em X_1 (ou equivalentemente, X_0). Figura extraída de Siqueira et al. [SLT$^+$08]. 15

4.1 Passos fundamentais em processamento de imagens digitais. Figura extraída de Gonzalez e Woods [GW02]. 17

4.2 Exemplos de limiarização. Imagem original à esquerda e após a limiarização com limiar = 80 à direita. 20

4.3 Exemplo de segmentação por limiarização: (a) Imagem com valores reais de intensidade. (b) Imagem binária obtida por limiarização. Figura extraída de Carvalho (1999). 23

4.4 Ilustração gráfica da M-semisegmentação cuja existência é garantida pelo Teorema 1. Figura extraída de Carvalho, Herman e Kong [CHK05]. . . . 26

4.5 Exemplo de segmentação *fuzzy* em imagem de ressonância magnética do cérebro humano – Original à esquerda e a 3-segmentação à direita. 30

5.1 Ilustração das definições $\mathcal{N}(p)$, $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$. Os pontos pertencentes aos conjuntos $\mathcal{N}(p)$, $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$ são representados pelos círculos preenchidos. Figura extraída de Siqueira et al. [SLT+08]. 34

5.2 Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente uma única instância de $C1$. Os pontos a e b representam os dois pontos de *background* de (D, $X^{(i-1)}$) que pertencem a instância de $C1$. Figura extraída de Siqueira et al. [SLT+08]. 35

5.3 Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente duas instâncias de $C1$. Os pontos do *foreground* e do *background* de (D, $X^{(i-1)}$) que pertencem às duas instâncias de $C1$ estão representadas como círculos brancos e pretos, respectivamente. Figura extraída de Siqueira et al. [SLT+08]. . . . 36

5.4 Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente três instâncias de $C1$. Os pontos do *foreground* e do *background* de (D, $X^{(i-1)}$) que pertencem às três instâncias de $C1$ estão representadas como círculos brancos e pretos, respectivamente. Figura extraída de Siqueira et al. [SLT+08]. 36

5.5 Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente uma única instância de $C2$. Os pontos a e b representam os pontos de *background* de (D, $X^{(i-1)}$) que pertencem a instância de $C2$. Figura extraída de Siqueira et al. [SLT+08]. 37

5.6 Outra ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente uma única instância de $C2$. Os pontos a, b, c, d, e e f representam os pontos de *background* de (D, $X^{(i-1)}$) que pertencem a instância de $C2$. Figura extraída de Siqueira et al. [SLT+08]. 37

5.7 Fatias de um volume do cérebro humano simulado pelo BrainWeb. Visão do plano sagital, coronal e axial do volume, da esquerda para a direita. . 41

5.8 Fatias Sagitais de dois volumes distintos de MRI. 42

5.9 Imagens binárias resultantes da segmentação *fuzzy* dos volumes BrainWeb 1 (sem ruído, à esquerda) e BrainWeb 6 (com nível de ruído de 9%, à direita). 42

5.10 Imagem original da fatia axial do volume BrainWeb 1 (à esquerda), imagem resultante bem-composta gerada pelo Ran3DWC (ao centro), imagem resultante bem-composta gerada pelo Ran3DWCFS (à direita). . . . 43

5.11 Imagem original da fatia axial do volume BrainWeb 1 (à esquerda), imagem resultante bem-composta gerada pelo Det3DWC (ao centro), imagem resultante bem-composta gerada pelo Det3DWCFS (à direita). 43

5.12 Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à quantidade de pontos modificados em cada volume da base Brain. 48

5.13 Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à quantidade de pontos modificados em cada volume da base BrainWeb. 49

5.14 Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à divisão do total dos graus de pertinência pela quantidade de pontos modificados (N/S) em cada volume da base Brain. 50

5.15 Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à divisão do total dos graus de pertinência pela quantidade de pontos modificados (N/S) em cada volume da base BrainWeb. 50

Lista de Tabelas

5.1 Descrição das escolhas pertencentes a $B(q)$ para cada um dos quatro casos mutuamente exclusivos (e suas subclasses) nos quais $\mathcal{I}(q)$ contém uma instância de C1 ou C2: denominação do caso (primeira coluna); número de instâncias de C1 em $\mathcal{I}(q)$ (segunda coluna); número de instâncias de C2 em $\mathcal{I}(q)$ (terceira coluna); as escolhas $\in B(q)$ (quarta coluna); e o número da figura que ilustra o referido caso (quinta coluna). Tabela extraída de Siqueira et al. [SLT+08]. 38

5.2 Apresentação dos volumes do cérebro simulados pelo BrainWeb e seus respectivos parâmetros. 42

5.3 Resultados obtidos pela execução do algoritmo Ran3DWC com os volumes de imagens de ressonância magnética Brain. 44

5.4 Resultados obtidos pela execução do algoritmo Ran3DWC com os volumes de imagens de ressonância magnética realística BrainWeb. 44

5.5 Resultados obtidos pela execução do algoritmo Ran3DWCFS com os volumes de imagens de ressonância magnética Brain. 45

5.6 Resultados obtidos pela execução do algoritmo Ran3DWCFS com os volumes de imagens de ressonância magnética realística BrainWeb. 45

5.7 Resultados obtidos pela execução do algoritmo Det3DWC com os volumes de imagens de ressonância magnética Brain. 45

5.8 Resultados obtidos pela execução do algoritmo Det3DWC com os volumes de imagens de ressonância magnética realística BrainWeb. 46

5.9 Resultados obtidos pela execução do algoritmo Det3DWCFS com os volumes de imagens de ressonância magnética Brain. 46

5.10 Resultados obtidos pela execução do algoritmo Det3DWCFS com os volumes de imagens de ressonância magnética realística BrainWeb. 46

Capítulo 1

Introdução

A representação de objetos geométricos através de imagens tridimensionais (3D) tem sido amplamente utilizada em diversas aplicações, tais como análise automática de amostras biológicas e sistemas de auxílio ao diagnóstico médico (*Computer-Aided Diagnosis* – CAD).

Uma imagem digital binária 3D consiste de uma matriz tridimensional de *voxels* (*volume elements*). Pode-se dizer que um *voxel*, em uma imagem 3D, corresponde a um *pixel* (*picture element*) em uma representação bidimensional. Cada *voxel* é representado por uma função $f(x, y, z)$, onde x, y e z denotam as coordenadas espaciais e $f(x, y, z)$ corresponde ao brilho (ou níveis de cinza) da imagem no ponto (x, y, z).

Sendo assim, nessas imagens, os objetos reais são representados como conjuntos finitos e discretos de *voxels*, resultantes de um processo de amostragem e quantização. Na computação visual, este processo é chamado de digitalização e é naturalmente realizado por dispositivos específicos como *scanners* para tomografias computadorizadas ou câmeras CCD (*Charge-Coupled Devices*).

Após a digitalização, tem-se o processo de segmentação da imagem que consiste em dividi-la em suas partes ou objetos constituintes, com o objetivo de simplificar e/ou mudar a representação da referida imagem para facilitar a sua análise [GW02]. Uma técnica bastante conhecida para segmentar uma imagem é através da escolha de um limiar T. Neste caso, os pontos que apresentam intensidade acima do referido limiar são classificados como pertencentes a um determinado objeto da imagem. A partir de então, atribui-se o valor 1 (um) a cada ponto pertencente ao objeto (*foreground*) e 0 (zero) aos demais pontos da imagem, correspondendo ao fundo da imagem (*background*).

Desta forma, no processo de reconstrução, cada ponto assinalado por 1 é identificado como um cubo (*voxel*) centrado no ponto. A união destes *voxels* formam um subconjunto

de $\mathbb{R}^3$, denominado de análogo contínuo do *foreground*, os quais podem ser vistos como uma representação "contínua" de um ou mais objetos representados por uma imagem multivalorada.

Sempre que uma imagem digital 3D é usada para representar objetos geométricos em aplicações práticas, espera-se que o análogo contínuo do *foreground*, correspondente a qualquer objeto na imagem, exiba propriedades fundamentais do objeto. Particularmente, espera-se que a topologia do análogo contínuo do *foreground* seja a mesma que a topologia do objeto. Contudo, devido a vários fatores relacionados ao processo de digitalização, a segmentação e o processo de reconstrução podem não conseguir produzir um análogo contínuo do *foreground* correto [LCG98, SK05]. Além disso, a topologia do objeto pode não ser conhecida ou não estar disponível para a aplicação, o que tornaria impossível a correção da topologia do análogo contínuo do *foreground*.

Em várias aplicações, os únicos objetos geométricos pertencentes à imagem são *sólidos*. Denominam-se *sólidos*, os objetos que representam subconjuntos fechados e limitados de $\mathbb{R}^3$, cuja fronteira seja uma superfície. Siqueira et al. [SLT+08] definem que se uma imagem tridimensional representa corretamente um objeto sólido, então esta imagem deve ser bem-composta. Desta forma, a fronteira do análogo contínuo do *foreground* representando o sólido deve ser uma superfície [Lat97]. Caso não seja, conclui-se que a topologia do análogo contínuo esteja incorreta. Assim, conclui-se que uma imagem binária 3D é considerada bem-composta se, e somente se, a união das faces compartilhadas pelos *voxels* do *foreground* e do *background* da referida imagem é uma superfície em $\mathbb{R}^3$.

Latecki [LCG98] provou que se a imagem digital não satisfaz a propriedade de ser bem-composta, então o processo de digitalização que a originou não preservou a topologia do objeto real. Se não for fornecida qualquer informação que permita derivar a topologia correta, qualquer tentativa de modificar a topologia do análogo contínuo torna-se um exercício de adivinhação.

Todavia, apesar de não podermos ter certeza de que a imagem bem-composta resultante seja verdadeiramente a correta, existem vantagens práticas em modificar uma imagem tridimensional a fim de torná-la bem-composta, conforme veremos adiante.

Imagens bem-compostas se beneficiam de propriedades topológicas muito interessantes, as quais nos permitem simplificar e otimizar a velocidade de algoritmos comumente usados na computação gráfica, processamento de imagens e visão computacional. Dentre estas propriedades topológicas, podemos citar que, numa imagem 3D bem-composta, a fronteira de toda componente conexa é uma superfície de Jordan [Lat97] e existe somente um tipo de componente conexa, visto que as componentes 6-, 14-, 18- e 26-conexas são iguais. Isto implica que, para estas imagens, a escolha do *foreground* e do *background*

não será crítica para o resultado de uma análise subseqüente [Lat97]. Desta forma, estas propriedades possibilitam que diversos algoritmos se tornem mais simples, quando a imagem de entrada é bem-composta.

Em particular, algoritmos usados na área de processamento de imagens para extrair superfícies e afinar imagens binárias podem ser simplificados e executados mais rapidamente se a imagem de entrada for bem-composta. Além disso, algoritmos para computar a curvatura (discreta) da superfície e extrair superfícies triangulares adaptativas que atuem diretamente na imagem binária, podem ser aplicados a estas imagens, evitando assim a necessidade de conversão da representação por imagem para outro tipo de representação, tal como malhas poligonais [SLT+08].

Estas vantagens têm motivado o desenvolvimento de algoritmos para reparar imagens bi e tridimensionais que não sejam bem-compostas, os chamados algoritmos reparadores [Lat98, SLT+08], que têm o objetivo de modificar uma imagem digital binária (que não é bem-composta) a fim de produzir uma imagem bem-composta. Um algoritmo reparador poderá ser usado sempre que uma imagem digital 3D representando um sólido não for bem-composta e nenhuma informação que nos permitirá encontrar a topologia correta do sólido for fornecida.

Contudo, para que estes algoritmos sejam realmente úteis na prática, eles devem produzir uma imagem bem-composta fazendo somente poucas modificações na imagem binária de entrada, de modo que a imagem bem-composta resultante não diferirá muito da referida imagem de entrada.

Diante disso, Siqueira et al. [SLT+08] desenvolveram um algoritmo reparador que tem a habilidade de gerar uma imagem 3D bem-composta com as mesmas dimensões de uma imagem tridimensional qualquer. O referido algoritmo percorre a imagem de entrada e muda iterativamente a cor de pontos do *background*, tornando-os pontos do *foreground*, para eliminar as configurações críticas encontradas. Contudo, este algoritmo não incorpora nenhuma informação adicional que possa ser utilizada nesta tomada de decisão. Portanto, a escolha de quais pontos do *background* serão modificados é aleatória.

Basicamente, o algoritmo proposto por Siqueira et al. [SLT+08] busca minimizar o número de pontos (do *background*) cuja cor é modificada. Contudo, nesta abordagem, não é considerada a "afinidade" desses pontos em relação ao *foreground* da imagem de entrada. Isto implica que a solução que minimiza o número de pontos modificados nem sempre pode ser a mais adequada na prática. Sendo assim, o ideal seria ter uma solução que modificasse a cor dos pontos com mais "afinidade" com o *foreground*.

O grau de afinidade dos pontos de imagem qualquer pode ser obtido, naturalmente, com a utilização de uma segmentação *fuzzy*. A segmentação *fuzzy* consiste em determinar

para cada elemento da imagem um grau de pertinência (no intervalo de zero a um) indicando a certeza dele pertencer (ou não) a um determinado objeto que, acredita-se, esteja contido na imagem [CGHK99, HC01, CHK05].

Diante do exposto, a presente dissertação propõe dois algoritmos reparadores de imagens binárias 3D, um aleatório e um determinístico. Nestes, a seleção de quais pontos de *background* virão a pertencer ao *foreground* da imagem resultante baseia-se nos graus de pertinência (afinidade) dos referidos pontos, resultante da segmentação *fuzzy* realizada no passo anterior. Desta forma, garante-se que um subconjunto dos pontos escolhidos pelos algoritmos em qualquer iteração seja um com a menor afinidade com o *background* dentre todas as escolhas possíveis.

1.1 Contribuições

As investigações realizadas nesta dissertação tiveram como objetivo principal desenvolver algoritmos reparadores que incorporassem informações adicionais no processo de tomada de decisão de quais pontos terão suas cores mudadas, durante a reparação topológica em imagens binárias 3D. Mais especificamente, informações oriundas do processo de segmentação *fuzzy*.

Para tanto, foram desenvolvidos dois algoritmos reparadores, um aleatório e um determinístico. Ambos são capazes de fazer reparos topológicos em imagens binárias 3D, produzindo imagens bem-compostas similares às imagens originais. A idéia fundamental por trás de ambos algoritmos é, assim como o de algoritmo proposto em Siqueira et al. [SLT$^+$08], mudar iterativamente a cor atribuída a alguns pontos da imagem de entrada de 0 (*background*) para 1 (*foreground*) até a imagem se tornar bem-composta. Contudo, os pontos cujas cores são mudadas pelos nossos algoritmos são escolhidos de acordo com seus valores de afinidade *fuzzy* em relação ao *foreground*, resultante do método de segmentação *fuzzy*, desenvolvido por Carvalho, Herman e Kong [CHK05].

Foram também realizados experimentos a partir de imagens de ressonância magnética do cérebro humano e análise dos resultados obtidos. Nesta fase, buscou-se avaliar a eficácia dos algoritmos propostos realizando a comparação dos resultados adquiros, por exemplo, a quantidade total de pontos modificados da imagem de origem e o somatório dos graus de pertinência destes, com os obtidos pelo algoritmo aleatorizado de Siqueira et al. [SLT$^+$08]. Os resultados obtidos se mostraram bastante satisfatórios.

1.2 Organização do trabalho

Para um melhor entendimento do escopo do assunto, organizamos o trabalho como se segue. No Capítulo 2, são apontados alguns trabalhos relacionados ao assunto de imagens bem-compostas, além da descrição do funcionamento do algoritmo presente em [SLT+08]. O Capítulo 3 apresenta alguns conceitos básicos sobre topologia digital e imagens digitais binárias bem-compostas.

O Capítulo 4 apresenta a segmentação *fuzzy* de imagens digitais e mostra o algoritmo rápido de segmentação *fuzzy* simultânea de múltiplos objetos, que será utilizado para segmentar as imagens de entrada e, desta forma, retornar o grau de pertinência de cada ponto. É imprescindível lembrar que este grau de pertinência será usado pelos nossos algoritmos reparadores a fim de escolher quais pontos serão modificados durante a geração da imagem bem-composta.

O Capítulo 5 descreve a incorporação da informação da segmentação *fuzzy* no algoritmo de reparo em [SLT+08], assim como o algoritmo determinístico. Também é realizada uma análise nos resultados obtidos pelos experimentos com imagens de ressonância magnética do cérebro humano realizados a partir do algoritmo de Siqueira et al. [SLT+08] e dos algoritmos reparadores aqui desenvolvidos. Finalmente, o capítulo 6 expõe as considerações finais e sugestões para trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Latecki, Eckhardt e Rosenfeld [LER95] propuseram em seu trabalho uma definição formal de uma classe especial de subconjuntos de imagens digitais binárias 2D, conhecida por imagens bem-compostas. Eles caracterizaram estas imagens pela ausência de determinada vizinhança $2X2$ dos pontos da imagem, denominada configuração crítica, apresentada na Figura 2.1. Sendo assim, provaram que estas configurações críticas podem ser detectadas localmente.

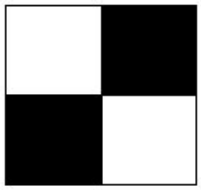

Figura 2.1: Configuração crítica de uma imagem 2D não bem-composta. Os quadrados pretos representam pontos do *foreground* e os brancos, pontos do *background*.

Posteriormente, Latecki [Lat97] definiu e analisou as propriedades das imagens 3D bem-compostas e as caracterizou em termos das ausência de certas vizinhanças $2X2$ e $2X2X2$ de pontos da imagem, que correspondem às configurações críticas em três dimensões, conforme demonstrada na Figura 3.3.

Latecki, Conrad e Gross [LCG98] demonstraram que um processo de digitalização preservando a topologia de um objeto r-*regular* deve resultar em uma imagem 2D bem-composta. Além disso, derivaram condições relacionadas às propriedades de um objeto r-regular ao tamanho da grade (do inglês, *grid*) do dispositivo de amostragem que garantem que o objeto real e sua imagem digital sejam topologicamente equivalentes. Consequentemente, se estas condições forem satisfeitas pelo processo de digitalização, logo a topologia do objeto pode ser corretamente recuperada da imagem pelo processo de segmentação

e reconstrução da imagem. Contudo, estas condições dependem de que a topologia do objeto seja conhecida a priori, o que é bastante difícil de acontecer na prática.

Stelldinger and Köthe [SK05] provaram que as condições propostas por Latecki, Conrad e Gross [LCG98] não são suficientes para garantir que a topologia do objeto r-regular 3D (i.e., um sólido) possa ser corretamente recuperada da imagem durante o processo de segmentação e reconstrução. Posteriormente, Stelldinger, Latecki e Siqueira [SLS07] propuseram condições suficientes para reconstruir corretamente a topologia de um sólido de uma imagem digital binária 3D. Porém, estas condições apresentam as mesmas limitações práticas das condições expostas por Latecki, Conrad e Gross [LCG98].

Latecki [Lat98] foi o primeiro a sugerir a idéia de modificar uma imagem binária a fim de produzir uma imagem bem-composta, bem como a propor o primeiro algoritmo para produzir uma imagem bem-composta 2D de uma imagem digital binária 2D. Contudo, este algoritmo não se aplicava às imagens binárias 3D.

Posteriormente, Rosenfeld, Kong e Nakamura [RKN98] introduziram um operador de imagens, denominado *simple deformation*, que pode ser usado para produzir imagens bem-compostas 2D e 3D. Contudo, a imagem resultante em 2D e 3D tem 9 e 27 vezes mais pontos que a imagem binária de entrada[1], respectivamente.

Além disso, diversos algoritmos propostos na literatura [MFB+95, MKAE00, FLD01, SL01, HXBNP02, HXP03, BK03, BP05, SGF05] estão relacionados a vários resultados que consideram a segmentação e reconstrução correta da superfície do córtex cerebral humano.

O córtex cerebral é a maior parte do cérebro humano e sua reconstrução correta a partir de imagens de ressonância magnética (RM) é um importante objetivo na medicina e neurociência. Embora o córtex cerebral seja altamente ”enrrugado”, ele apresenta somente poucos milímetros de espessura e é comparado a uma folha fina dobrada. É composto de substância cinzenta (GM) que situa-se envolto pelo líquido cerebrospinal (ou líquido cefalorraquidiano – CSF) e envolve a substância branca do cérebro (WM).

Se a abertura do tronco cerebral é artificialmente fechada, a superfície do córtex apresenta a topologia de uma esfera. Devido a ruídos da imagem, iluminação não uniforme e a natureza altamente convolucionada do córtex cerebral, torna-se difícil produzir uma representação precisa e topologicamente correta. Sendo assim, o maior problema topológico resultante de uma digitalização ou segmentação incorreta é a presença de alças (do inglês, handles) na superfície reconstruída [HXBNP02].

[1]Este algoritmo insere fatias (do inglês, *slices*) extras na imagem original [RKN98].

Algoritmos para segmentação e reconstrução de modelos de córtex cerebral podem ser subdivididos em dois tipos, a saber, aqueles que incorporam algum tipo de mecanismo de preservação da topologia durante o processo de segmentação [MFB+95, MKAE00, HXBNP02, BK03], e aqueles que não incorporam [FLD01, SL01, HXBNP02, BP05, SGF05, KG01]. Os algoritmos que preservam a topologia tipicamente iniciam com uma superfície cuja topologia é conhecida e, então, a deformam iterativamente de modo que ele se aproxime o máximo possível da geometria da superfície do córtex cerebral.

Existem dois problemas principais relacionados aos algoritmos de segmentação que preservam a topologia. Primeiro, eles podem levar a imprecisões geométricas. Segundo, eles também necessitam de uma inicialização perto do córtex. Então, a correção da topologia do córtex é frequentemente necessária, seja antes ou depois da inicialização, que pode ser realizada usando intensidade local, probabilidades a priori e informação geométrica sem considerar a topologia [BP05].

Existem vários algoritmos destinados a corrigir a topologia de uma imagem de RM do cérebro anteriormente segmentada. A lógica deles baseia-se em identificar as alças e então escolher entre cortar uma alça (*foreground*) ou preencher um orifício (*background*), conforme exposta na Figura 2.2. Alguns algoritmos assumem que as alças estão localizadas nas partes mais finas da região de interesse da imagem, e faz suas decisões ao modificar o mínimo possível a região ou uma superfície triangular aproximando a região da fronteira [SL01, HXBNP02]. Embora eles frequentemente conduzam a resultados precisos, as correções topológicas podem não ser as melhores. Outros algoritmos tem alcançado melhores resultados ao integrar informações estatísticas ou geométricas durante o processo de decisão [FLD01, KG01, SGF05].

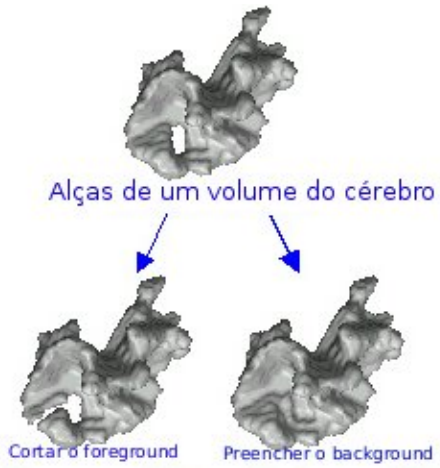

Figura 2.2: Alça feita a partir de um volume da substância branca do cérebro humano e o resultado da aplicação da técnica do corte desta alça e do preenchimento do ofício.

Recentemente, Bazin e Pham [BP05] propuseram um algoritmo que corrige a topologia de uma imagem anteriormente segmentada usando uma função da distância de preser-

vação da topologia. Este algoritmo modifica a cor associada a pontos da imagem a fim de preservar a topologia de uma iso-superfície definida pela função da distância. As mudanças topológicas são detectadas pela análise dos registros da função dos pontos críticos.

A despeito das vantagens dos algoritmos supracitados, eles assumem que a topologia do objeto, representado por uma dada imagem, é conhecida a priori. Isso não é uma característica desejável, visto que a topologia do objeto pode não ser conhecida ou não estar disponível para a aplicação, o que tornaria impossível a correção da mesma.

Siqueira, Latecki e Gallier [SLG04] propuseram o primeiro algoritmo reparador capaz de gerar uma imagem 3D bem-composta com o mesmo tamanho da imagem digital de entrada, sem necessitar de qualquer informação adicional sobre a topologia do objeto representado por esta imagem. Para tanto, o referido algoritmo, encontrando configurações críticas, muda iterativamente o valor da cor de certos pontos do *background* da imagem, de maneira a tornarem-se pontos do *foreground* numa imagem de saída. Este processo é inicialmente executado na imagem de entrada ($\mathbb{G}$, X), e então na imagem obtida pela execução anterior. O algoritmo pára quando a imagem resultante da mais recente execução da operação é bem-composta ($\mathbb{G}$, X'). A escolha de quais pontos de *background* serão transformados em *foreground* é aleatória.

Não há garantias que a imagem bem-composta resultante do algoritmo proposto por Siqueira, Latecki e Gallier [SLG04] seja a mesma de uma obtida por um processo de digitalização que preserva a topologia. Entretanto, como citado no Capítulo 1, existem várias aplicações na computação gráfica, processamento de imagens e visão computacional que se beneficiam ao lidar com imagens bem-compostas. Além do mais, a maior parte dos algoritmos de correção da topologia acima citados restringem-se a corrigir a topologia de imagens de RM do córtex cerebral humano, enquanto o algoritmo aleatorizado de Siqueira, Latecki e Gallier [SLG04] ultrapassa esta limitação podendo ser aplicado a qualquer tipo de imagens de ressonância magnética.

Siqueira et al. [SLT$^+$08] apresentaram uma nova versão deste algoritmo aleatorizado e introduziram uma extensão do mesmo para reparar imagens multivaloradas digitais 3D. Contudo, a lógica que norteia o funcionamento do algoritmo apresentado por Siqueira, Latecki e Gallier [SLG04] permaneceu a mesma. Logo, a escolha de quais pontos de *background* seriam transformados em *foreground* é também aleatória.

Por sua vez, o algoritmo porposto pelo presente trabalho estende a lógica do algoritmo apresentado por Siqueira et al. [SLT$^+$08], associando informações resultantes da aplicação do algoritmo rápido para segmentação *fuzzy*, desenvolvido por Carvalho, Herman e Kong [CHK05], como auxílio na tomada de decisão de quais pontos do *background* venham a ser modificados (tornando-se pontos do *foreground*) durante a execução de um

algoritmo reparador. Isso possibilitará que o algoritmo proposto retorne imagens bem-compostas mais fiéis às imagens de entrada.

Da mesma forma que o algoritmo de Siqueira et al. [SLT+08], o nosso algoritmo gera imagens 3D bem-compostas com o mesmo tamanho das imagens de entrada de qualquer tipo de imagens de resonância magnética. Contudo, também pode ser aplicado a qualquer tipo de imagens digitais 3D.

Embora a idéia fundamental do algoritmo proposto pelo presente trabalho seja similar ao de Siqueira et al. [SLT+08], para este último a escolha de quais pontos de *background* serão transformados em *foreground* é aleatória. Em nosso algoritmo, o fato de aplicar informações da própria segmentação da imagem, resultou em um algoritmo determinístico.

Capítulo 3

Imagens Digitais Binárias 3D Bem-Compostas

Eckhardt e Latecki [EL94] definem topologia digital como sendo o campo de pesquisa que estuda as propriedades topológicas dos subconjuntos do plano digital, os chamados conjuntos digitais. Define-se plano digital $\mathbb{Z}^2$ (ou volume $\mathbb{Z}^3$) como sendo o conjunto de todos os pontos no plano $\mathbb{R}^2$ (ou $\mathbb{R}^3$) que apresentam coordenadas inteiras. No processamento digital de imagens, cada conjunto digital corresponde ao conjunto de *pixels* (ou *voxels*) que representam um determinado objeto constituinte da imagem 2D (ou 3D). Desta forma, a topologia digital estuda as propriedades topológicas deste conjunto de pixels (ou voxels) da imagem que correspondem às do objeto real.

Se o processo de digitalização preservar a topologia do objeto real, a imagem resultante será realmente bem-composta [LER95]. Para exemplificar, considere que a digitalização seja definida a partir de uma grade de células quadradas de diâmetro d e que os pontos pretos nesta grade representam o *foreground* e os brancos representam o *background*, conforme Figura 3.1.

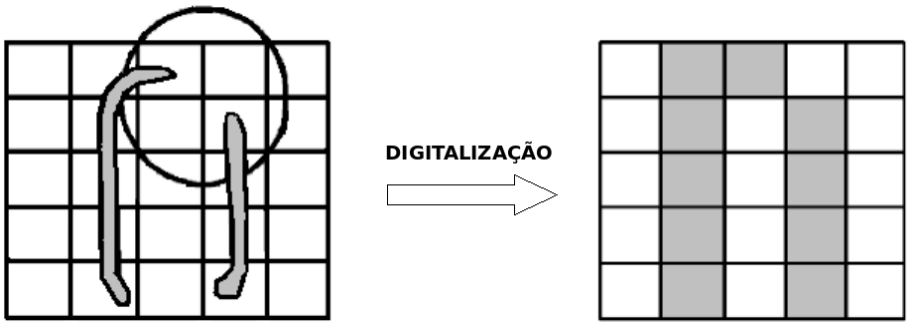

Figura 3.1: Um conjunto desconectado pode ter uma digitalização conectada. Figura extraída de Latecki, Eckhardt e Rosenfeld [LER95].

Durante o processo de digitalização, cada célula da grade será um *pixel* preto se, e somente se, contiver um ponto preto. Como demonstrado na Figura 3.1, um conjunto desconectado S de *pixels* pretos podem resultar em uma digitalização conectada. Perceba que no círculo de raio d, a interseção de S com duas células adjacentes diagonalmente são desconectados por duas outras células adjacentes diagonalmente, os quais não interceptam S. Caso esta situação ocorra, o processo de digitalização não preservará a topologia e a imagem resultante da digitalização não será bem-composta, conforme ilustrado na referida figura.

3.1 Definições Básicas

Nesta seção, são introduzidos brevemente alguns conceitos básicos de topologia digital importantes para o entendimento do que vêm a ser imagens bem-compostas. Uma discussão mais detalhada acerca deste assunto pode ser encontrada em [Her98] e [Lat97].

Como é comum em Geometria Digital, interpreta-se $\mathbb{Z}^3$ como sendo um conjunto de pontos de $\mathbb{R}^3$ com coordenadas inteiras. Denomina-se de *voxel*, ou análogo contínuo, um cubo em $\mathbb{R}^3$ centrado em um ponto de $\mathbb{Z}^3$ e cujas faces são paralelas às coordenadas planas. Sendo assim, um conjunto digital tridimensional (i.e., um subconjunto finito de $\mathbb{Z}^3$) pode ser identificado pela união desses voxels, o que permite uma correspondência simples e intuitiva entre pontos em $\mathbb{Z}^3$ e cubos em $\mathbb{R}^3$. Considerando que esta correspondência desempenha um importante papel neste trabalho, ela será definida formalmente a seguir.

O *análogo contínuo* CA(p), também denominado *voxel*, de um ponto $p \in \mathbb{Z}^3$ é uma função

$$\text{CA} : \mathbb{Z}^3 \to \mathcal{P}(\mathbb{R}^3), \tag{3.1}$$

onde $\mathcal{P}(\mathbb{R}^3)$ denota o conjunto das partes de $\mathbb{R}^3$. Por sua vez, o análogo contínuo de um conjunto digital X (i.e., $X \subseteq \mathbb{R}^3$) é definido como

$$\text{CA}(X) = \cup\{\text{CA}(x) : x \in X\}. \tag{3.2}$$

Neste caso, a função CA pode ser vista como sendo

$$\text{CA} : \mathcal{P}(\mathbb{Z}^3) \to \mathcal{P}(\mathbb{R}^3), \tag{3.3}$$

onde $\mathcal{P}(\mathbb{Z}^3)$ é o conjunto das partes de $\mathbb{Z}^3$. A Figura 3.2 mostra um exemplo de um análogo contínuo CA(X) de um conjunto $X = \{a, b, c, d\}$, consistindo de quatro pontos a, b, c e d, de $\mathbb{Z}^3$.

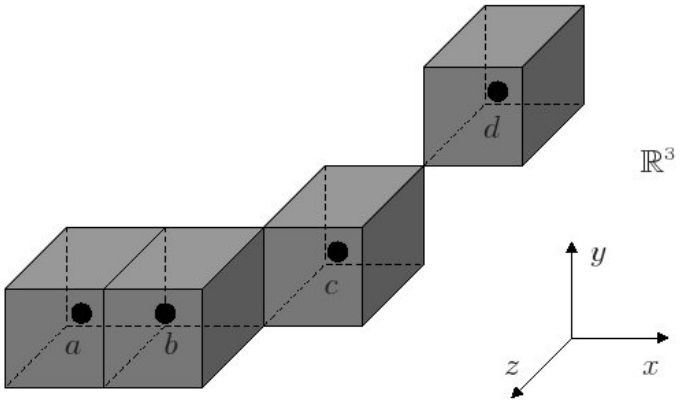

Figura 3.2: Análogo contínuo do conjunto $X = \{a, b, c, d\} \subset \mathbb{Z}^3$. Figura extraída de Siqueira et al. [SLT+08].

Dois pontos distintos $p, q \in \mathbb{Z}^3$ são denominados *adjacentes por face* se CA(p) e CA(q) compartilham uma face, ou em outras palavras, se duas das coordenadas de p e q são iguais e as terceiras diferem por 1. Por outro lado, dois pontos distintos $p, q \in \mathbb{Z}^3$ são considerados *adjacentes por aresta* se CA(p) e CA(q) compartilham uma aresta, mas não uma face (i.e., CA(p) $\cap$ CA(q) é um segmento de linha), ou equivalentemente, se uma das coordenadas de p e q é a mesma e as outras duas coordenadas diferem por 1. E, finalmente, dois pontos distintos $p, q \in \mathbb{Z}^3$ são ditos *adjacentes por vértice* se CA(p) e CA(q) compartilham um vértice, mas não uma aresta (i.e., CA(p) $\cap$ CA(q) é um ponto simples), ou equivalentemente, se todas as três coordenadas de p e q diferem por 1. Exemplificando, temos na Figura 3.2, os pontos a e b, que são adjacentes por face. Os pontos b e c são adjacentes por aresta, mas não são adjacentes por face. E, finalmente, os pontos c e d são adjacentes por vértice, mas não são adjacentes por faces nem por aresta.

Dois pontos são ditos 6-adjacentes (vizinhança de 6) se eles são adjacentes por face; 18-adjacentes (vizinhança de 18), se eles são adjacentes por face ou por aresta; e 26-adjacentes (vizinhança de 26), se eles são adjacentes por face, aresta ou vértice. Por exemplo, de acordo com a Figura 3.2, os pontos a e b são 6-adjacentes, 18-adjacentes e 26-adjacentes. Os pontos b e c são 18-adjacentes e 26-adjacentes, mas não 6-adjacentes. E, finalmente, os pontos c e d são 26-adjacentes, mas não 18-adjacentes nem 6-adjacentes. Um conjunto $X \subset \mathbb{Z}^3$ é k-adjacente para um ponto $p \in \mathbb{Z}^3$ se existir $q \in X$ tal que p e q sejam k-adjacentes, onde k = 6, 18, 26. $\mathcal{N}_k(p)$ denota o conjunto contendo $p \in \mathbb{Z}^3$ e todos os pontos k-adjacentes a p e $\mathcal{N}_k^*(p)$ denota $\mathcal{N}_k(p) \setminus \{p\}$, onde k = 6, 18, 26. $\mathcal{N}_{26}(p)$ é também referenciado por $\mathcal{N}(p)$ e é denominado *vizinhança* de p, enquanto que $\mathcal{N}_{26}(p) \setminus \{p\}$ é referenciado por $\mathcal{N}^*(p)$.

Seja A um conjunto qualquer, e $\rho \in \{6, 18, 26\}$. Para qualquer p e q pertencentes ao conjunto A, a sequência $\langle x^{(0)}, \cdots, x^{(k)} \rangle$ de elementos de A, onde $k \in \mathbb{Z}$, com $k \geq 0$, é denominada ρ-caminho em A conectando p a q se $x^{(0)} = p$, $x^{(k)} = q$, e $x^{(i-1)}$ é ρ-

adjacente a $x^{(i)}$, onde $i \in \mathbb{Z}$ e $1 \leq i \leq k$. O valor k é denominado comprimento do caminho. Particulamente, são denominados de *caminhos triviais*, os ρ-caminhos de comprimento zero; por exemplo, $\langle p \rangle$. Se existir um ρ-caminho em A conectando p a q, então p é ρ-conexo em A a q. Por exemplo, tomemos A como o conjunto de pontos na Figura 3.2. Então, o ponto a é 6-conexo ao ponto b, 18-conexo ao ponto c, e 26-conexo ao ponto d. Logo, existe um 26-caminho de comprimento 3 em A conectando a a d.

Seja A um conjunto qualquer, e $\rho \in \{6, 18, 26\}$. É dito que A é ρ-conexo se, para todo p, $q \in A$, existir um ρ-caminho em A conectando p a q. Se B é um subconjunto ρ-conexo máximo de A, então diz-se que B é uma componente ρ-conexo, ou simplesmente ρ-componente de A. Desta forma, considerando a Figura 3.2, o conjunto é A é 26-conexo e $B = \{a, b, c\}$ é um 18-componente de A.

A fronteira de um análogo contínuo bdCA(X) de um conjunto digital $X \subseteq \mathbb{Z}^3$ é definida como sendo a união de um conjunto de faces, cada uma das quais é uma face comum a um *voxel* de CA(X) e a um *voxel* não pertencente a CA(X), sendo assim, pertencente ao análogo contínuo do complemento de X. Logo,

$$bd\text{CA}(X) = bd\text{CA}(X^c), \tag{3.4}$$

onde $X^c = \mathbb{Z}^3 \setminus X$. Formalmente, a fronteira de um conjunto digital $X \subseteq \mathbb{Z}^3$ pode ser definida como sendo o conjunto de pares

$$bd(X) = \{(p, q) : p \in X \text{ e } q \in X^c \text{ e } p \text{ é face-adjacente a } q\}. \tag{3.5}$$

Deste modo, uma imagem digital 3D $\mathcal{I} : D \rightarrow V$ é uma função de um subconjunto D, denominado de grade, de pontos em $\mathbb{Z}^3$ para um subconjunto V de $\mathbb{R}$, denominado de cores. Uma imagem digital binária 3D é uma imagem $\mathcal{I} : D \rightarrow V$ no qual o conjunto de cores V é um conjunto binário $\{0, 1\}$. Usualmente, uma imagem binária $\mathcal{I}$ é denotada pelo par (D, X), onde D é a grade de $\mathcal{I}$ e X é o conjunto $\{p \in D \mid \mathcal{I}(p) = 1\}$, nos quais são tratados como pontos do *foreground* da imagem. Por sua vez, o conjunto X^c é o complemento de X e comumente denominado de pontos do *background* de (D, X), onde $X^c = D \setminus X = \{p \in D \mid \mathcal{I}(p) = 0\}$. Para simplificar, denotaremos o *foreground* X por X_1 e o *background* X^c por X_0.

3.2 Imagens Digitais 3D Bem-compostas

Assumindo que X ou o seu complemento $X^c = \mathbb{Z}^3 \setminus X$ é um conjunto digital, Latecki [Lat97] define que uma imagem digital 3D ($\mathbb{Z}^3$, X) é dita bem-composta se bdCA(X) é uma superfície em $\mathbb{R}^3$.

Um subconjunto $S \subset \mathbb{R}^3$ é denominado de *superfície topológica*, ou simplesmente superfície, se cada ponto $p \in S$ tem uma vizinhança aberta homeomorfa a um disco aberto do Espaço Euclidiano bidimensional, E^2. Por exemplo, se (D, X) é uma imagem binária cujo *foreground* $X_1 = X$ é o conjunto de pontos $\{a, b, c, d\}$ na Figura 3.2, então (D, X) não é bem-composta, visto que qualquer ponto da aresta CA(b) $\cap$ CA(c) , assim como, o ponto CA(c) $\cap$ CA(d) não têm nenhuma vizinhança que é homeomorfa a $\mathbb{R}^2$.

Latecki [Lat97] concluiu que a propriedade de uma imagem ser bem-composta é, na verdade, equivalente à ausência de determinadas configurações críticas envolvendo *voxels* de CA(X) e CA(X^c) que contém uma face em bdCA(X), citadas a seguir:

- Seja A um conjunto qualquer de quatro pontos de D. Considera-se A como sendo uma configuração crítica 1 ($C1$) em (D, X) se dois pontos de A pertencem a X_1, os outros dois pontos pertencem a X_0, os dois pontos em X_1 (respectivamente, X_0) são adjacentes por aresta, e os voxels dos quatro pontos compartilham uma aresta.
- Seja A um conjunto qualquer de oito pontos de D. Considera-se A como sendo uma configuração crítica 2 ($C2$) em (D, X) se dois (respectivamente, seis) pontos de A pertencem a X_1, os outros seis (respectivamente, dois) pontos pertencem a X_0, os dois pontos em X_1 (respectivamente, X_0) são adjacentes por vértice, e os voxels dos oito pontos compartilham um vértice.

Desta maneira, uma imagem digital 3D (D, X) é bem-composta se, e somente se, não existir configurações críticas de $C1$ e $C2$ em (D, X).

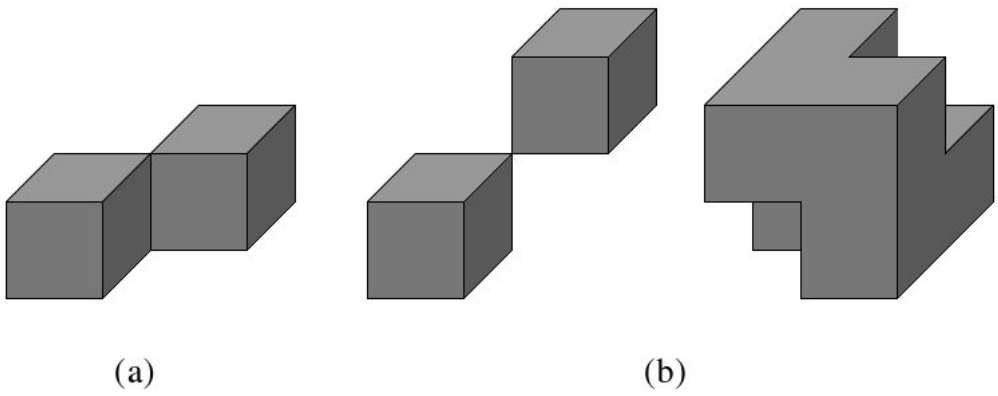

Figura 3.3: Tipos de configurações críticas: (a) Configuração Crítica 1. (b) Configuração Crítica 2. Para um melhor entendimento, ambas as imagens mostram apenas voxels em X_1 (ou equivalentemente, X_0). Figura extraída de Siqueira et al. [SLT+08].

De acordo com Siqueira et al. [SLT+08], existe um algoritmo simples para decidir se uma dada imagem binária digital 3D (D, X) é bem-composta: para cada subconjunto Y

de quatro (respectivamente, oito) pontos de D, cujos *voxels* compartilham uma aresta (respectivamente, um vértice), basta verificar se Y é uma instância de $C1$ (respectivamente, $C2$) em (D, X), conforme as definições de configurações críticas expostas acima.

Capítulo 4

Segmentação Fuzzy de Imagens Digitais

Processar uma imagem consiste na sua manipulação através de técnicas que objetivam a extração da informação nela contida [GW02], visando uma melhoria de informação visual para interpretação humana ou aplicação em processos como, por exemplo, aprendizagem de máquina e reconhecimento de padrões.

Durante o processamento de uma imagem digital, a sua análise se resume no processo de descoberta, identificação e compreensão dos padrões que são relevantes na realização de tarefas baseadas em imagens. Uma das principais etapas deste processo consiste na segmentação, durante a qual são extraídos os objetos e outras entidades utilizadas para o processamento subsequente da imagem, conforme apresentados na Figura 4.1.

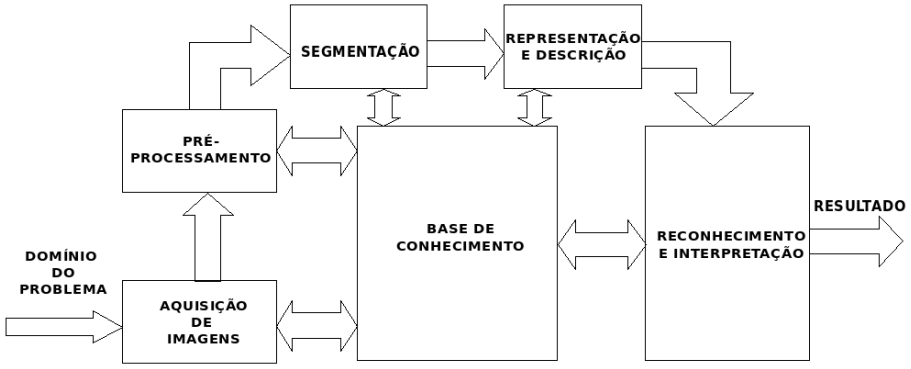

Figura 4.1: Passos fundamentais em processamento de imagens digitais. Figura extraída de Gonzalez e Woods [GW02].

Conforme mostra a Figura 4.1, qualquer aplicação de processamento de imagens pode ser dividida em passos fundamentais. O primeiro passo é *aquisição da imagem*, que consiste em adquirir uma imagem digital a partir de um sensor de imageamento. Caso a saída deste não se encontre na forma digital, um conversor analógico-digital a digitaliza. A seguir, o *pré-processamento* envolve o realce das características desejáveis na imagem para serem usadas nas demais etapas do processamento.

Posteriormente, a *segmentação* subdivide uma imagem em suas partes ou objetos constituintes. Algoritmos de segmentação permitem achar diferenças entre dois ou mais objetos, e distinguir as partículas umas das outras e do fundo da imagem. Tal distinção permitirá a um programa detectar *pixels* (respectivamente, *voxels*) contíguos e agrupá-los em regiões. Esse procedimento será detalhadamente descrito na Seção 4.2.

Já a *representação* consiste em transformar o resultado da segmentação em uma forma adequada para o subsequente processamento computacional. A *descrição*, por sua vez, separa as diferentes classes de objetos na imagem. Por fim, a etapa de *reconhecimento* rotula cada objeto a partir dos dados provenientes do descritor, enquanto que a *interpretação* envolve a atribuição de significado aos objetos reconhecidos.

Os algoritmos de segmentação podem ser classificados como manuais, automáticos e semi-automáticos. Nos algoritmos de segmentação semi-automáticos, a intervenção manual é utilizada para fornecer pontos característicos da estrutura a ser segmentada ou para delinear uma região onde ela é encontrada. Já na segmentação automática, não há intervenção humana. Contudo, pode haver intervenção para eventuais correções da segmentação obtida.

Alguns algoritmos de segmentação geram segmentações que utilizam a lógica *fuzzy* [LL99], que está mais de acordo ao pensamento humano onde os valores das variáveis possuem pesos que mudam de acordo com o contexto. O algoritmo de segmentação *fuzzy* de imagens utilizado nesta dissertação é usado com sucesso em imagem que apresentam sombras, iluminação não uniforme e/ou ruídos [HC01, CHK05], conforme veremos na Seção 4.3.

4.1 Definições Básicas

Matematicamente, uma imagem é definida [MV89] como uma função $f(x, y)$, bidimensional, válida em uma certa região. Normalmente, esta região é um subconjunto do plano espacial, e a função pode assumir apenas valores reais não-negativos e limitados. Por exemplo, em fotografias monocromáticas, a função $f(x, y)$ representa a intensidade da luz refletida por objetos, onde x e y denotam as coordenadas espaciais do plano e o valor de $f(x, y)$ refere-se a intensidade (brilho) da imagem naquele ponto, capturada através de algum processo físico [GW02].

Em uma imagem digital, os valores assumidos pela função $f(x, y)$ geralmente são números inteiros, e a região em que a função é definida constitui um arranjo retangular de pontos, denominado grade. Cada elemento (ou célula) dessa grade é chamado de *pixel*

(abreviação de *picture element*), sendo caracterizado por sua posição na imagem e pelo seu valor.

Em processamento de imagens, a obtenção de uma imagem digital a partir de uma imagem contínua é realizada através de um processo de digitalização, que envolve duas etapas, que são a amostragem e a quantização. Na amostragem, determina-se quais serão as dimensões do arranjo retangular. Na quantização, por sua vez, determina-se um valor que represente digitalmente o conteúdo da região amostrada.

Como se trata de um valor digital, o valor de um pixel está sempre limitado, de acordo com a quantidade de níveis de intensidade que podem ser percebidos pelo dispositivo de aquisição de imagens [GW02]. Esta limitação é normalmente medida em bits por pixel. Valores usuais são 1, 2, 4, 6 e 8 bits por pixel, gerando, respectivamente, 2, 4, 16, 64 ou 256 intensidades diferentes para cada pixel. Em imagens monocromáticas, utiliza-se uma escala denominada de escala de cinza, que varia linearmente do preto (com menos intensidade) ao branco (com maior intensidade), para representar a variação do valor dos pixels.

4.2 Segmentação de Imagens

Gonzales e Woods [GW02] define o processo de segmentação de imagens como sendo a subdivisão de uma imagem em suas partes ou objetos constituintes, de acordo com suas propriedades intrínsecas, a saber, nível de cinza, contraste ou propriedades texturais. O nível até o qual essa subdivisão deve se realizada, assim como a técnica utilizada, depende do problema que está sendo resolvido. Sendo assim, o processo de segmentação deve parar quando os objetos de interesse na aplicação tiverem sido isolados. Devido à existência de várias soluções "corretas", a segmentação de imagens é um problema mal-posto.

Por exemplo, numa análise de imagens de ressonância magnética do cérebro humano, o interesse reside, entre outras coisas, em identificar suas estruturas anatômicas a fim de detectar possíveis anomalias. O primeiro passo do processo de segmentação desta imagem pode resultar na extração do cérebro da caixa craniana, identificando tecidos que pertencem ao cérebro (estrutura craniana) e tecidos que não pertencem ao cérebro. Um passo posterior, mais detalhado, pode ser a segmentação, e consequente identificação, dos três tecidos cerebrais, a saber: a substância cinzenta (GM), a substância branca (WM) e o líquido cerebrospinal (CSF). Finalmente, pode-se delinear estruturas anatômicas específicas do cérebro como o córtex, o cerebelo, o hipotálamo e os ventrículos.

Os algoritmos para segmentação de imagens em tons de cinza (também denominadas, imagens monocromáticas) são geralmente baseados em uma das duas propriedades básicas de valores de níveis de cinza: descontinuidade ou similaridade [GW02]. No primeiro caso, a abordagem consiste em particionar a imagem baseado em mudanças bruscas nos níveis de cinza, como por exemplo, detecção de pontos isolados, linhas e bordas na imagem. Enquanto que, no segundo caso, procura-se detectar regiões com tons de cinza semelhantes. Neste, as principais abordagens baseiam-se em limiarização, crescimento de regiões e divisão e fusão de regiões.

Segundo Gonzalez e Woods [GW02], a limiarização (ou *thresholding*) é a mais simples abordagem para segmentação de imagens. Esta técnica consiste em selecionar um limiar T que será utilizado para classificar a qual grupo (objeto ou *background*) um *pixel* deverá pertencer. Para definir qual o valor do limiar T a ser utilizado, pode ser utilizada uma informação global da imagem através do histograma das intensidades das imagens monocromáticas, ou uma informação local somente de uma parte da imagem.

Desta forma, uma imagem limiarizada $g(x, y)$ é definida como:

$$g(x,y) = \begin{cases} 1 & \text{se } f(x,y) > T, \\ 0 & \text{se } f(x,y) \leq T, \end{cases} \tag{4.1}$$

onde $f(x, y)$ é o nível de cinza do ponto (x, y). Portanto, *pixels* rotulados com 1 (um), ou qualquer outro nível de cinza conveniente, correspondem aos objetos, enquanto que aqueles rotulados com 0 (zero) correspondem ao fundo (ou *background*). A Figura 4.2 mostra um exemplo de limiarização em uma imagem com 255 níveis de cinza.

Também é possível estabelecer múltiplos limiares objetivando-se o isolamento de mais de duas regiões de interesse, usando uma *limiarização multiníveis*. Enfim, para cada n valores de limiar, a imagem é divida em $n + 1$ regiões.

Figura 4.2: Exemplos de limiarização. Imagem original à esquerda e após a limiarização com limiar = 80 à direita.

4.2.1 Segmentação Orientada a Regiões

A segmentação orientada a regiões divide a imagem em partes com a finalidade de identificar regiões distintas [Fac93, MV89]. Gonzalez e Woods [GW02] citam uma definição clássica da segmentação orientada a regiões, descrita a seguir.

Seja R a região completa da imagem. Pode-se considerar a segmentação como o processo de particionar R em n regiões $R_1, R_2, \ldots, R_n$, tal que:

(a) $\bigcup_{i=1}^{n} R_i = R$

(b) R_i é uma região conexa, $i = 1, 2, \ldots, n$,

(c) $R_i \cap R_j = \varnothing$ para todo i e j, $i \neq j$,

(d) $P(R_i) =$ VERDADEIRO para todo $i = 1, 2, \ldots, n$, e

(e) $P(R_i \cup Rj) =$ FALSO para $i \neq j$,

em que $P(R_i)$ é um predicado lógico sobre os pontos do conjunto R_i e $\varnothing$ é o conjunto vazio.

A condição (a) indica que a segmentação deve ser completa, ou seja, cada *pixel* deve pertencer a uma região. A condição (b) indica que cada região deve ser conexa. Por sua vez, a condição (c) assegura que nenhum ponto da imagem pode pertencer a mais de uma região ao mesmo tempo. Já a condição (d) trata das propriedades que devem ser satisfeitas pelos *pixels* para pertecerem a uma determinada região segmentada. E, finalmente, a condição (e) garante que as regiões R_i e R_j são diferentes de acordo com o sentido do predicado lógico P.

A segmentação orientada a regiões pode ser realizada através das seguintes técnicas: crescimento de regiões, divisão de regiões, fusão de regiões, ou combinando as técnicas anteriores [GW02].

4.2.1.1 Crescimento de Regiões por Agregação de *Pixels*

Segundo Gonzalez e Woods [GW02], o procedimento denominado de *crescimento de regiões por agregação de pixels* agrupa *pixels* ou sub-regiões em regiões maiores, a partir de pontos sementes (*seed pixels*). Sendo assim, no início do processo, tem-se um conjunto de pontos sementes e, a partir deles, as regiões vão sendo construídas em torno de cada

ponto semente pela junção de *pixels* vizinhos que possuam propriedades similares, como nível de cinza, textura ou cor.

A escolha dos conjuntos de pontos sementes para cada região é uma parte essencial na segmentação por crescimento de regiões por agregação de *pixels*. A seleção das sementes está diretamente relacionada ao conhecimento que o usuário (ou programa) tem sobre o conteúdo da imagem, e também pode influenciar diretamente no cálculo das propriedades que serão utilizadas no processo de expansão das regiões. Deste modo, a seleção de pontos sementes que representem adequadamente as regiões de interesse, bem como a seleção de propriedades apropriadas para a inclusão de pontos nas várias regiões durante o processo de crescimento apresentam-se como passos importantes durante a aplicação da técnica de crescimento de regiões.

Gonzales e Woods [GW02] mencionam que a seleção de um ou mais pontos iniciais pode frequentemente se basear na natureza do problema. Além disso, eles afirmam que a seleção do critério de similaridade depende não somente do problema em consideração, mas também do tipo de imagem a ser segmentada, pois cada tipo de imagem possui características que influenciam na escolha do critério de similaridade, como na segmentação de imagens monocromáticas, coloridas ou de profundidade (range image).

4.3 Segmentação *Fuzzy* de Imagens

A limiarização não é um método apropriado de segmentação nos casos em que a característica que distingue o objeto de interesse do *background* é alguma propriedade textural, ao invés do exato valor atribuído aos *pixels*, como por exemplo, imagens contendo sombras, iluminação não uniforme e/ou ruídos [HC01]. A presença dessas interferências em uma imagem requer dos algoritmos de segmentação um esforço adicional durante o reconhecimento dos objetos presentes na mesma.

A Figura 4.3 exemplifica a dificuldade de se encontrar um limiar que consiga separar objetos do *background* em uma imagem com iluminação não uniforme. A Figura 4.3(a) é uma imagem sintética definida em uma grade hexagonal e que possui um retângulo na sua parte superior, no qual a intensidade da iluminação aumenta lentamente da esquerda para a direita, e um *background* ruidoso. Como mostrado em Carvalho et al. [CGHK99], é impossível encontrar um limiar T capaz de produzir um resultado que segmente completamente o retângulo do *background*. Entretanto, a segmentação *fuzzy* tem sido aplicada com sucesso nesses casos.

A segmentação *fuzzy* é uma técnica que determina para cada elemento da imagem um grau de pertinência (entre zero e um) indicando a certeza dele pertencer (ou não)

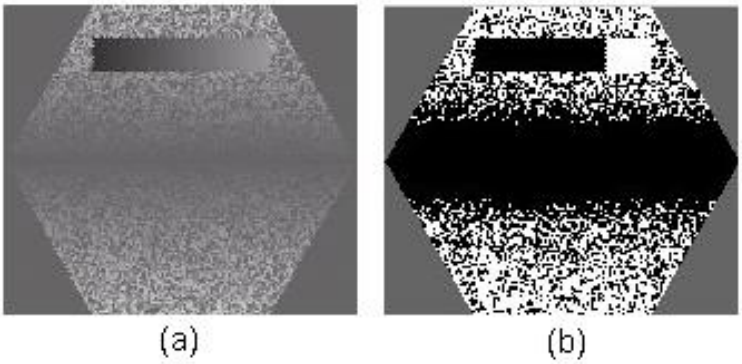

Figura 4.3: Exemplo de segmentação por limiarização: (a) Imagem com valores reais de intensidade. (b) Imagem binária obtida por limiarização. Figura extraída de Carvalho (1999).

a um determinado objeto (que acredita-se estar contido na imagem) [CHK05]. Essas informações de pertinência possibilitam que o algoritmo segmente a imagem tomando uma decisão mais flexível acerca da classificação de cada elemento.

O Algoritmo de Segmentação *Fuzzy* (MOFS – *Mult-Object Fuzzy Segmentation*) implementado por Herman e Carvalho [HC01] utiliza o conceito de conectividade *fuzzy*, proposto por Rosenfeld [Ros79], e se baseia no trabalho desenvolvido por Udupa e Samarakera [US96], mas é generalizado para espaços digitais arbitrários de acordo com a teoria proposta por Herman [Her98].

4.3.1 Segmentação *Fuzzy* Simultânea de Múltiplos Objetos

Herman e Carvalho introduziram o algoritmo MOFS [HC01] que atribui, simultaneamente, a cada elemento da imagem um grau de pertinência em relação a múltiplos objetos, que acredita-se estarem contidos na imagem. Este algoritmo é um método semi-automático de segmentação por crescimento de regiões no qual o usuário pode escolher um ou mais *spels* (do inglês, *spacial elements*) sementes para representar os objetos que se deseja segmentar.

Denomina-se espaço digital o par ordenado (V, π), onde V é um conjunto arbitrário e π é uma relação binária simétrica em V, tal que V é um conjunto conectado por π. Os elementos de V são referidos por *spels*. Sendo o conjunto V finito, o par ordenado (V, π) pode ser interpretado como um grafo conectado [Har69], com V sendo o conjunto de nós e π, o conjunto de arcos. Os spels de um conjunto arbitrário V podem representar, dentre outras coisas, pontos em um plano, *pixels* em uma imagem ou *voxels* em um volume.

Uma afinidade *fuzzy* do *spel* é uma função $\psi : V^2 \rightarrow [0, 1]$. Esta atribui valor entre 0 e 1 para cada par ordenado de *spels*, tal que

(i) para todo $c \in V$, $\psi(c, c) = 1$, e

(ii) para todo c, $d \in V$, $\psi(c, d) = \psi(d, c)$.

No decorrer da execução do algoritmo de Carvalho, Herman e Kong [CHK05], os elementos do conjunto V são associados a um objeto utilizando o conceito *fuzzy*. Sendo assim, a partir da função de afinidade, determina-se, para cada *spel*, o grau de pertinência em relação ao objeto ao qual foi associado. Para fazer a análise de cada *spel* e encontrar o grau de pertinência que esse *spel* tem com cada objeto da imagem é utilizado o conceito de conectividade *fuzzy* [CGHK99], conforme definida a seguir.

Uma *corrente* de $c^{(0)}$ a $c^{(K)}$ corresponde a uma sequência arbitrária de $\langle c^{(0)}, \ldots, c^{(K)} \rangle$ de *spels* distintos. Considera-se cada par ordenado $(c^{(k-1)}, c^{(k)})$ de *spels* consecutivos como sendo um *elo* na corrente e $\psi(c^{(k-1)}, c^{(k)})$, a ψ-força do elo. A ψ-força da corrente $\langle c^{(0)}, \ldots, c^{(K)} \rangle$ é definida como sendo a ψ-força do seu elo mais fraco, ou seja, o mínimo de $\psi(c^{(k-1)}, c^{(k)})$, com $1 \leq k \leq K$.

Consequentemente, a conectividade *fuzzy* de um par de *spels* (c, d) é a ψ-força da corrente mais forte de c a d. No caso de uma determinada corrente possuir somente um *spel*, a sua ψ-força é definida como sendo 1.

As funções ψ que calculam as afinidades entre *spel* são as funções de afinidade *fuzzy*, que indicam o quão estreita é a relação entre os *spels* em questão. Em outras palavras, ele indica o grau de confiança de que, após a segmentação, os dois *spels* deverão pertencer a mesma classe. Logo, a atribuição de um valor 0 indica que o *spel* definitivamente não pertence ao objeto e 1 indica que ele definitivamente pertence. Deste modo, a função $\psi(c, d)$ retorna um número real entre 0 (zero) e 1 (um), denominado de conectividade *fuzzy* entre c e d. Esta abordagem gera um conceito de *conjunto fuzzy*, definido em [PM86]. Neste contexto, o conjunto *fuzzy* refere-se ao conjunto de pares conectados e o grau de afinidade do par (c, d), neste conjunto, é a conectividade *fuzzy* entre c e d.

Durante a definição da afinidade *fuzzy* do *spel* devem ser consideradas, entre outras coisas, a proximidade dos *spels* analisados, a similaridade dos valores atribuídos a eles na imagem que está sendo segmentada e a variância destes valores atribuídos aos vizinhos imediatos dos dois *spels* [CGHK99].

Para a segmentação *fuzzy* de imagens, a função de afinidade de um determinado objeto pode ser automaticamente definida baseando-se em propriedades estatísticas dos elos dentro de regiões identificadas pelos usuários como pertencendo ao objeto de interesse.

Para formalizar tal particionamento *fuzzy*, foi criado o conceito de M-semisegmentação (onde M é o número de objetos). Uma M-semisegmetação de V é uma função σ que mapeia cada $c \in V$ em um vetor de $(M+1)$-dimensões $\sigma^c = (\sigma_0^c, \sigma_1^c, \ldots, \sigma_M^c)$, onde

1. $\sigma_0^c \in [0,1]$ (isto é, σ_0^c está entre 0 e 1),
2. para cada $m(1 \leq m \leq M)$, o valor de σ_m^c é zero ou σ_0^c,
3. pelo menos para um $m(1 \leq m \leq M)$, $\sigma_m^c = \sigma_0^c$,

onde σ_m^c informa o grau de pertinência do *spel* c pertencer ao m-ésimo objeto, e σ_0^c é igual a $\max_{1 \leq m \leq M} \sigma_m^c$. Se, para todo *spel* c, σ_0^c for positivo, é dito que σ é uma M-segmentação.

Um conjunto U ($\subseteq V$) é dito *ψ-conectado* se, para todo par ordenado de *spels* em U, existe uma corrente em U de ψ-força positiva do primeiro ao segundo *spel*.

Por sua vez, um grafo *M-fuzzy* é um par (V, Ψ), onde V é um conjunto finito e não-vazio e Ψ = $(\psi_1, \ldots, \psi_M)$ com ψ_m (para $1 \leq m \leq M$) sendo uma função de afinidade *fuzzy*. Um grafo *M-fuzzy semeado* é uma tripla (V, Ψ, ν) onde (V, Ψ) é um grafo *M-fuzzy* e $\nu = (V_1, \ldots, V_M)$ onde $V_m \subseteq V$ para $1 \leq m \leq M$. Desta forma, um grafo *M-fuzzy semeado* $(V, (\psi_1, \ldots, \psi_M), (V_1, \ldots, V_M))$ é considerado conectado se:

1. o conjunto V é *ϕ_Ψ-conectado*, onde $\phi_\Psi(c,d) = \min_{1 \leq m \leq M} \psi_m(c,d)$ para todo c, $d \in V$, e
2. $V_m \neq 0$, para pelo menos um m, $1 \leq m \leq M$.

Para uma M-semisegmentação σ de V e para $1 \leq m \leq M$, uma corrente $\langle c^{(0)}, \ldots, c^{(K)} \rangle$ é definida como sendo uma *σm-corrente* se $\sigma_m^{c^{(k)}} > 0$, para $1 \leq k \leq K$. Além disso, para $W \subseteq V$ e $c \in V$, será usado $\mu_{\sigma,m,W}(c)$ para denotar a σm-corrente de ψ_m-força máxima de um *spel* em W para c. Se tal corrente não existir, essa força será igual a zero.

Teorema 1 *Se (V, Ψ, ν) é um grafo M-fuzzy semeado (onde $\Psi = (\psi_1, \ldots, \psi_M)$ e $\nu = (V_1, \ldots, V_M)$ então*

1. *existe uma M-semisegmentação σ de V com as seguintes propriedades: para todo $c \in V$, se para $1 \leq m \leq M$*

$$s_n^c = \begin{cases} 1, & \textit{se } c \in V_n, \\ \max_{d \in V}(\min(\mu_{\sigma,n,Vn}(d), \psi_n(d,c))), & \textit{caso contrário,} \end{cases} \tag{4.2}$$

então para $1 \leq m \leq M$

$$\sigma_m^c = \begin{cases} s_m^c, & \textit{if } s_m^c \geq s_n^c, \textit{ for } 1 \leq n \leq M, \\ 0, & \textit{caso contrário}; \end{cases} \tag{4.3}$$

2. *esta M-segmentação é única; e*

3. *ela é uma M-segmentação, desde que (V, Ψ, ν) seja conectado.*

As provas das partes 1 e 2 deste Teorema foram publicadas por Herman e Carvalho [HC01], enquanto que a prova da parte 3 foi publicada em Carvalho, Herman e Kong [CHK05]. A Figura 4.4 descreve graficamente o Teorema 1. Suponha, como na Figura 4.4, que c seja um *spel* arbitrário e que σ^d seja conhecido para os demais *spels* d. Então, para $1 \leq n \leq M$ (neste caso, M=3), o s_n^c da Equação 4.2 é a ψ_n-força máxima de uma corrente $\langle d^{(0)}, \cdots, d^{(L)}, c\rangle$ de um *spel* semente em V_n para c de modo que $\sigma_n^{d^{(l)}} > 0$ (ou seja, $d^{(l)}$ pertença ao n-ésimo objeto) para $0 \leq l \leq L$. Intuitivamente, o n-ésimo objeto só pode assegurar que c pertença a ele se, e somente se, s_n^c for máximo. Isto está de acordo com a Equação 4.3, que determina que σ_m^c tem um valor positivo somente para o objeto cujo valor de s_m^c é o máximo. Além disso, esta propriedade nos mostra como um *spel* pode se relacionar com vários objetos, desde que seja conhecido o σ para todos os outros *spels*: com a Equação 4.3 satisfeita, pode-se calcular os valores de s_m^c para um *spel* c utilizando a Equação 4.2. O Teorema 1 diz que existe uma, e somente uma, M-semisegmentação que satisfaz simultaneamente as proposições, e que esta M-semisegmentação é de fato uma M-segmentação se o grafo M-*fuzzy semeado* for conectado.

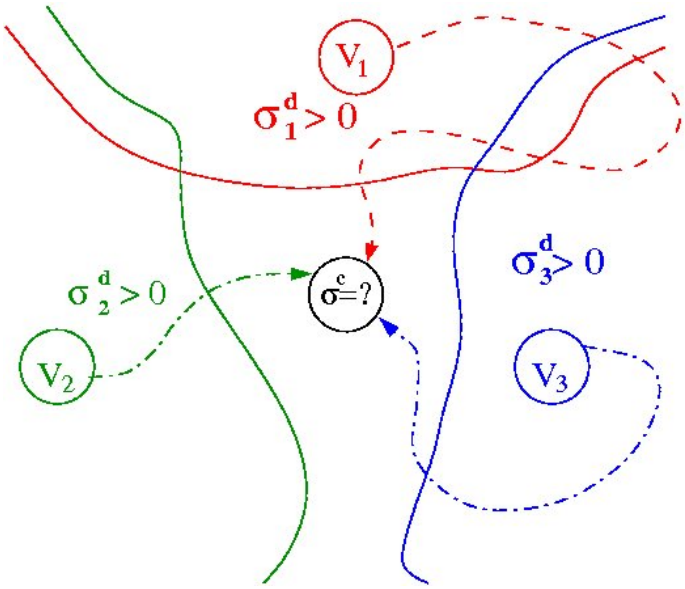

Figura 4.4: Ilustração gráfica da M-semisegmentação cuja existência é garantida pelo Teorema 1. Figura extraída de Carvalho, Herman e Kong [CHK05].

Na Seção 4.2.1, a condição **c** afirma que não existe interseção entre duas regiões adjacentes (R_i e R_j). Entretanto, na segmentação *fuzzy* descrita em Herman e Carvalho [HC01] e em Carvalho, Herman e Kong [CHK05], um dado *spel* c pode ter uma mesma afinidade em relação a duas ou mais correntes de dois ou mais objetos distintos, e deste modo, pertencer a dois ou mais objetos.

4.3.1.1 Algoritmo Rápido para Segmentação *Fuzzy*

Carvalho, Herman e Kong [CHK05] apresentaram uma implementação rápida do algoritmo MOFS original, publicado em Herman e Carvalho [HC01]. Esta nova versão pode ser utilizada em aplicações que necessitam de um tempo de resposta pequeno e cuja qualidade de segmentação não seja consideravelmente afetada por uma discretização dos valores gerados pela função de afinidade. O resultado deste algoritmo é uma M-segmentação σ, de acordo com o Teorema 1, mas usando um grafo M-*fuzzy* cujas funções de afinidade *fuzzy* podem assumir um número resumido de valores.

Nesta nova implementação, os valores de σ_0^c são armazenados em listas, e não em um *heap* binário, como na implementação do algoritmo MOFS de Herman e Carvalho [HC01]. Isso é justificado pelo fato de que em muitas aplicações a qualidade da segmentação *fuzzy* não é significativamente afetada se os valores de afinidade forem arredondados para, por exemplo, 3 (três) casas decimais. Suponha que R é um conjunto não vazio de afinidades *fuzzy* para uma classe particular de problemas e que ele é sempre um subconjunto de um conjunto A. Sendo K a cardinalidade do conjunto $A \cup \{1\}$ e seja $1 = a_1 > a_2 > \cdots > a_k > 0$ os elementos de A. Se for usado valores de afinidades arredondados para 3 casas decimais, logo $A = \{0.001, 0.002, \ldots, 0.999, 1.000\}$, de forma que $K = 1000$ e $a_k = 1.001 - k/1000$.

Algoritmo 1

1- Algoritmo MOFS - Versão rápida

1. ***for*** $c \in V$ ***do***

2. ***for*** $m \leftarrow 0$ ***to*** M ***do***

3. $\sigma_m^c \leftarrow 0$

4. ***for*** $m \leftarrow 1$ ***to*** M ***do***

5. ***for*** $c \in V_m$ ***do***

6. $\sigma_0^c \leftarrow \sigma_m^c \leftarrow 1$

7. $U[m][1] \leftarrow V_m$
8. ***for*** $k \leftarrow 2$ ***to*** K ***do***
9. $U[m][k] \leftarrow \emptyset$
10. ***for*** $k \leftarrow 1$ ***to*** K ***do***
11. ***for*** $m \leftarrow 1$ ***to*** M ***do***
12. ***while*** $U[m][k] \neq \emptyset$ ***do***
13. *remova um spel* d *do conjunto* $U[m][k]$
14. $C \leftarrow \{c \in V \mid \sigma_m^c < \min(a_k, \psi_m(d,c))$ ***and*** $\sigma_0^c \leqslant \min(a_k, \psi_m(d,c))\}$
15. ***while*** $C \neq \emptyset$ ***do***
16. *remova o spel* c *de* C
17. $t \leftarrow \min(a_k, \psi_m(d,c))$
18. ***if*** $\sigma_0^c < t$ ***then do***
19. *remova* c *de cada conjunto em* U *que satisfaça a condição*
20. ***for*** $n \leftarrow 1$ ***to*** M ***do***
21. $\sigma_n^c \leftarrow 0$
22. $\sigma_0^c \leftarrow \sigma_m^c \leftarrow t$
23. *insira* c *no conjunto* $U[m][l]$ *onde* l *é o inteiro tal que* $a_l = t$

Esta versão do algoritmo MOFS utiliza uma matriz de $M \times K$ denominada $U[m][k]$, que armazena os conjuntos dos nós que representam cada *spel*, onde M representa o número de objetos e K a cardinalidade do conjunto A. Cada conjunto $U[m][k]$ é representado por uma lista duplamente ligada de nós. Os nós de conjuntos diferentes que representam o mesmo *spel* são mantidos juntos numa lista encadeada, o que torna fácil remover um *spel* de todos os conjuntos que o contém. Uma lista de nós contém as coordenadas dos seus *spels* enquanto que os dados do *spel* c inclui um ponteiro para a lista encadeada dos nós que representam c.

O processo é iniciado (Passos 1 a 9) primeiramente zerando todos os σ_m^c, para cada *spel* $c \in V$ e $0 \leq m \leq M$. Então, para todo *spel* $c \in V_m$, c é adicionado em $U[m][1]$ e as variáveis σ_0^c e σ_m^c são atualizados para 1. Em seguida, os demais valores de $U[m][k]$, com $2 \leq k \leq K$, são zerados.

Após a inicialização, tem-se o laço principal que é repetido K vezes. Esta parte do algoritmo é responsável pela atualização da suposição atual do valor final de σ_m^c. O valor de σ_m^c é substituído por um valor maior se for encontrada uma σm-corrente de um *spel*

semente em V_m para c com ψ_m-força maior que o valor antigo, e ele será atualizado para zero, se descobrir que (para um $n \neq m$) existe um σm-corrente de um *spel* semente em V_n para c com ψ_n-força maior que o antigo valor de σ_m^c.

Nos testes realizados e apresentados por Carvalho, Herman e Kong [CHK05], a implementação rápida do algoritmo MOFS conseguiu apresentar bom êxito em relação ao algoritmo MOFS original. Por exemplo, para calcular uma 4-segmentação de um volume de tomografia computadorizada com 7.281.928 *spels* usando um computador equipado com um processador 1.7GHz Intel Xeon, a versão rápida do MOFS precisou apenas de $35s$ (aproximadamente, $5\mu s$ por *spel*), enquanto que o algoritmo original o fez em $249s$ (aproximadamente, $34\mu s$ por *spel*).

4.3.1.2 Exemplos de aplicação do MOFS

O algoritmo MOFS pode ser utilizado em diversas aplicações, tais como de clusterização [JMF99] e na segmentação de imagens, de um modo geral [COG6a, CON6b, HC01]. Nesta seção apresentaremos o resultado da aplicação da versão rápida do MOFS na segmentação de imagens 3D de ressonância magnética (MRI) do cérebro humano.

Durante a segmentação *fuzzy* de uma imagem qualquer, é necessário definir ψ_m e V_m com $1 \leqslant m \leqslant M$, em outras palavras, a função de afinidade e os conjuntos dos pixels sementes. Neste caso, o conjunto V_m é formado pelos pontos sementes e os seus 6 vizinhos de face, que pertencem ao m-ésimo objeto. Na versão rápida do algoritmo MOFS, as sementes são definidas pelo usuário através de cliques na imagem, definindo para cada clique, qual objeto (classe) aquele ponto pertence.

Para a definição de ψ_m para $1 \leqslant m \leqslant M$, leva-se em consideração que importantes características de um objeto são definidas pelos valores dos pixels da imagem em uma região e pela diferença desses valores na sua vizinhança [CGHK99]. Carvalho, Herman e Kong [CHK05] utilizaram o seguinte procedimento para calcular as funções de afinidade. Defina ψ_m como:

$$\psi_m(c,d) = \begin{cases} 0, & \text{se } c,d \text{ não forem adjacentes,} \\ [\rho_{g_m,h_m}(g) + \rho_{a_m,b_m}(a)]/2, & \text{caso contrário,} \end{cases} \tag{4.4}$$

onde g_m é a média e h_m é o desvio padrão da média das intensidades (valores dos *spels*) dos pares adjacentes por face dos *spels* em V_m (isto é, 6-adjacência), e a_m é a média e b_m é o desvio padrão da diferença absoluta das intensidades dos pares adjacentes por face dos *spels* em V_m (isto é, 6-adjacência). O valor de g refere-se a média e a é a diferença absoluta entre as intensidades de c e d. A função $\rho_{r,s}(x)$ é a função de densidade de

probabilidade gaussiana com média r e desvio padrão s multiplicada por uma constante para que o seu valor máximo seja 1.

Na Figura 4.5, temos um exemplo da aplicação do algortimo MOFS em uma imagem de ressonância magnética do cérebro humano. Neste caso, foi feita uma 3-segmentação para destacar as substâncias branca e cinzenta do restante do cérebro e do *background*.

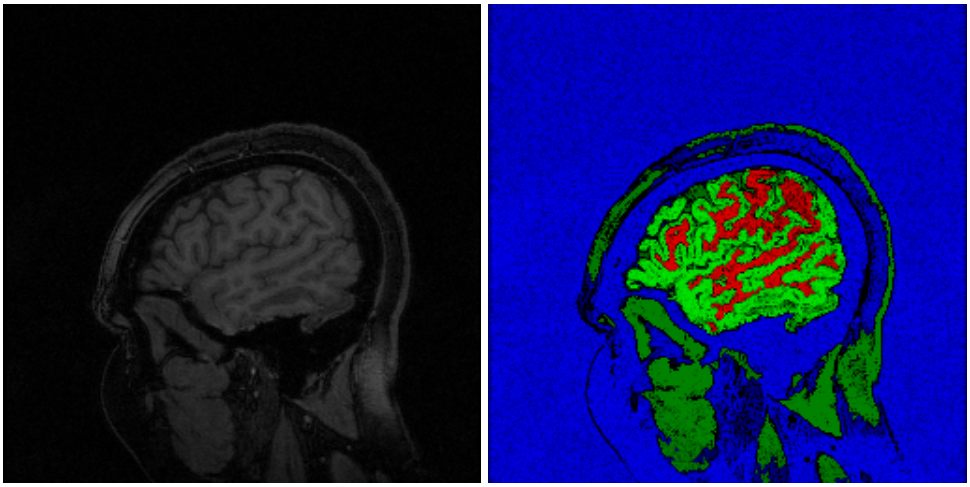

Figura 4.5: Exemplo de segmentação *fuzzy* em imagem de ressonância magnética do cérebro humano – Original à esquerda e a 3-segmentação à direita.

Capítulo 5

Reparação de Imagens Bem-Compostas Utilizando Mapas de Conectividade *Fuzzy*

Como exposto no Capítulo 1, a motivação inicial deste projeto surgiu da necessidade de associar alguma informação que auxiliasse a tomada de decisão do algoritmo reparador Ran3dwc de [SLT+08] na fase de escolha de quais pontos venham a ser modificados, a fim de transformar uma imagem de entrada em uma respectiva imagem bem-composta. A proposta inicial consistiu em adicionar funcionalidades ao algoritmo Ran3DWC de modo que ele passasse a utilizar informações resultantes do próprio processo de segmentação *fuzzy* da imagem de entrada, cujo algoritmo será apresentado na Seção 5.1.

Devido ao excelente êxito da versão rápida do algoritmo MOFS em testes realizados e apresentados por Carvalho, Herman e Kong [CHK05] e pela sua aplicação em imagens de um modo geral, optou-se por utilizá-lo para segmentar as imagens que serão tratadas pelo algoritmo reparador.

Conforme apresentado no Capítulo 4, sabe-se que, concluída a segmentação *fuzzy*, o MOFS retorna o mapa de conectividade *fuzzy* da imagem de entrada, o qual apresenta o grau de pertinência de cada ponto pertencente à imagem em relação a um número de objetos. Logo, propõe-se a utilização deste grau de pertinência para precisar qual(is) ponto(s) do *background*, envolvido(s) numa configuração crítica, tem(êm) a maior probabilidade (menor afinidade com o *background*) de pertencer ao *foreground*, durante o processo de geração de uma imagem bem-composta. Sendo assim, o uso do mapa de conectividade *fuzzy* garante que um subconjunto dos pontos escolhidos pelo algoritmo em qualquer iteração seja um com a menor afinidade com o *background* dentre todas as escolhas possíveis.

Posteriormente, foi desenvolvido um algoritmo reparador capaz de realizar reparos topológicos em imagens digitais binárias 3D. Embora a idéia fundamental deste algoritmo seja similar à de Siqueira et al. [SLT+08], apresentado no Capítulo 2, para este último, a escolha de quais pontos do *background* serão transformados em *foreground* é aleatória. No algoritmo determinístico implementado, exposto na Seção 5.2, essa escolha se baseia na quantidade de configurações críticas que possam vir a surgir com essa mudança e nos graus de pertinência dos referidos pontos, resultante da segmentação *fuzzy* anteriormente realizada.

5.1 Ran3DWCFS - Um Algoritmo Reparador Aleatório para Imagens Bem-Compostas utilizando Segmentação *Fuzzy*

A lógica do algoritmo aleatório para imagens bem-compostas utilizando segmentação *fuzzy* baseia-se de uma forma geral na implementação do algoritmo apresentado por Siqueira et al. [SLT+08], acrescentando apenas a informação sobre os graus de pertinência dos pontos da imagem, resultantes da segmentação *fuzzy*.

O referido algoritmo recebe como imagem de entrada (D, X) uma imagem segmentada pelo algoritmo MOFS [CHK05] e o respectivo grau de pertinência de cada ponto em relação ao objeto ao qual pertence; retornando uma imagem bem-composta (D, X') similar a de entrada. Para descobrir X', o algoritmo encontra um subconjunto P do *background* X_0 de (D, X) tal que $(D, X \cup P)$ é bem-composta, logo $X' = X \cup P$. Desta forma, P pode ser visto como um subconjunto (do *background*) com a menor afinidade com o *background* dentre todas as escolhas possíveis, cujas cores atribuídas serão mudadas de 0 para 1, a fim de produzir (D, X'). No contexto de aplicações práticas é importante que (D, X) e (D, X') sejam tão similares quanto possíveis.

A solução ótima para maximizar a similaridade entre (D, X) e (D, X') seria encontrar o menor subconjunto P de X_0, tal que $(D, X \cup P)$ seja bem-composta. Isto pode ser obtido, trivialmente, enumerando e testando todos os subconjuntos de X_0. Todavia, este procedimento tem complexidade de tempo exponencial no tamanho de G, representado por $\mid G \mid$, o que torna sua aplicação prática inaceitável.

A solução proposta em Siqueira et al. [SLT+08] apresenta complexidade de tempo linear ao buscar minimizar o subconjunto P de X_0. Entretanto, este algoritmo não considera a "afinidade" desses pontos em relação ao *foreground* da imagem de entrada. Isto implica que a solução que minimiza o número de pontos modificados nem sempre pode

ser a mais adequada na prática. Sendo assim, o ideal seria ter uma solução que modificasse a cor dos pontos com mais "afinidade" com o *foreground*. O algoritmo proposto no presente trabalho não garante encontrar o menor conjunto P, uma vez que ele busca encontrar um subconjunto P que apresenta a menor afinidade com o *background* (em outras palavras, que possui a menor soma dos graus de pertinência). Contudo, ele também apresenta complexidade de tempo linear em $\mid G \mid$.

Este algoritmo inicia atribuindo $P = 0$. Então, percorre todos os pontos de D procurando por instâncias de $C1$ e $C2$ em (D, X). No caso de nenhuma configuração crítica ser encontrada, o algoritmo termina com $X' = X$. Caso contrário, para cada instância de $C1$ e $C2$, o algoritmo compara os respectivos graus de pertinência dos pontos de *background* da respectiva configuração crítica. O ponto de $X_0 - P$ que possuir o menor grau de pertinência será inserido em P, até que a imagem $(D, X \cup P)$ torne-se bem-composta. Sempre que o algoritmo insere um ou mais pontos em P, ele elimina pelo menos uma configuração crítica de $C1$ ou $C2$ da imagem corrente, $(D, X \cup P)$. Contudo, a inserção de pontos pode também dar origem a outras instâncias de $C1$ e $C2$ em $(D, X \cup P)$, as quais serão eventualmente eliminadas da imagem resultante pelas inserções de outros pontos.

5.1.1 Eliminação das Configurações Críticas

A seguir, é exposta uma breve explanação de como ocorre a busca e eliminação das configurações críticas, apresentadas na Seção 3.2, pelo Algoritmo Aleatório para Imagens Bem-Compostas utilizando Segmentação *Fuzzy*.

Seja a imagem de entrada (D, X), onde $D = [d_{11}, d_{12}] \times [d_{21}, d_{22}] \times [d_{31}, d_{32}]$, onde $d_{i1}, d_{i2} \in \mathbb{Z}$ e $d_{i1} < d_{i2}$ para $1 \leq i \leq 3$, e, para qualquer ponto $p = (p_1, p_2, p_3) \in \mathbb{Z}^3$, temos

$$\begin{aligned}
\mathcal{N}_x(p) &= \{(p_1, y, z) \mid y \in \{p_2, p_2 + 1\} \text{ e } z \in \{p_3, p_3 + 1\}\}, \\
\mathcal{N}_y(p) &= \{(x, p_2, z) \mid x \in \{p_1, p_1 + 1\} \text{ e } z \in \{p_3, p_3 + 1\}\}, \\
\mathcal{N}_z(p) &= \{(x, y, p_3) \mid x \in \{p_1, p_1 + 1\} \text{ e } y \in \{p_2, p_2 + 1\}\},
\end{aligned}$$

e

$$\mathcal{N}(p) = \{p_1, p_1 + 1\} \times \{p_2, p_2 + 1\} \times \{p_3, p_3 + 1\}.$$

Esta definição encontra-se ilustrada na Figura 5.1.

Logo, uma imagem $\mathcal{I}$ pode ser definida como a seguir [SLT+08]:

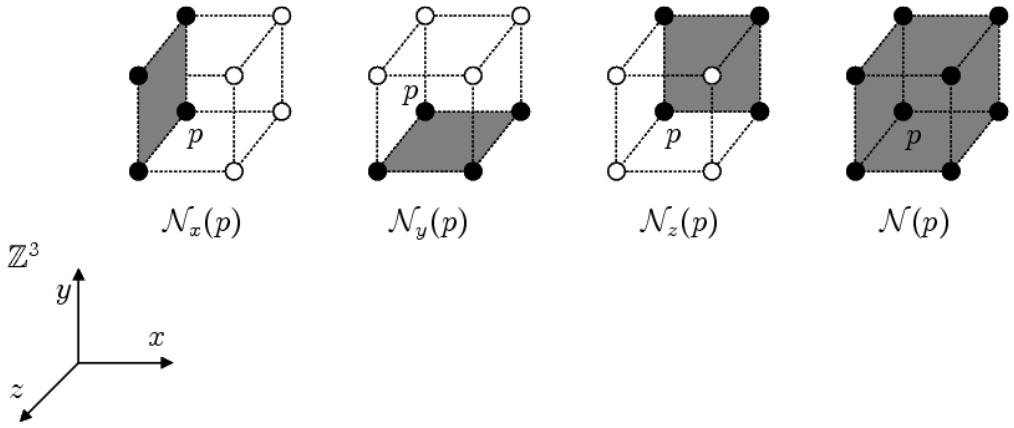

Figura 5.1: Ilustração das definições $\mathcal{N}(p)$, $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$. Os pontos pertencentes aos conjuntos $\mathcal{N}(p)$, $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$ são representados pelos círculos preenchidos. Figura extraída de Siqueira et al. [SLT+08].

$$\mathcal{I}(p) = \{\mathcal{N}_x(p), \mathcal{N}_y(p), \mathcal{N}_z(p), \mathcal{N}(p)\}.$$

É dito que $\mathcal{I}(p)$ contém uma instância de $C1$, se algum dos $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$ for uma instância de $C1$ em (D, X). Da mesma forma, se $\mathcal{N}(p)$ for uma instância de $C2$ em (D, X), diz-se que $\mathcal{I}(p)$ contém uma instância de $C2$. Considerando que cada instância de $C1$ consiste de quatro pontos em um dos $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$ e $\mathcal{N}_z(p)$ e que cada instância de $C2$ consiste de oito pontos em $\mathcal{N}(p)$, para qualquer $p \in D$, consequentemente conclui-se que, se pelo menos um de $\mathcal{N}_x(p)$, $\mathcal{N}_y(p)$, $\mathcal{N}_z(p)$ é uma instância de $C1$ em (D, X), então $\mathcal{N}(p)$ não pode ser uma instância de $C2$ em (D, X). Portanto, cada $\mathcal{I}(p)$ ou não contém configuração crítica, ou contém pelo menos uma instância de $C1$, ou contém exatamente uma instância de $C2$.

Estas observações acima conduzem a um procedimento simples para encontrar todas as instâncias de $C1$ e $C2$ numa imagem de entrada (D, X), como descrito a seguir.

Seja Q uma fila vazia. Para cada ponto $p \in D$, verifica-se se $\mathcal{I}(p)$ contém uma configuração crítica. Caso tenha, insere-se p em Q. Se Q permanece vazia então (D, X) não tem configurações críticas e o algoritmo termina com $X' = X$. Do contrário, o algoritmo inicia o passo de eliminação das configurações críticas. Perceba que um ponto $p \in Q$ se, e somente se, $\mathcal{I}(p)$ contém uma instância de $C1$ ou $C2$.

O passo da eliminação das configurações críticas consiste em um *loop* que é executado até que a fila Q seja vazia. Por enquanto, assuma que Q tornar-se-á vazia depois de um número finito, n, de iterações do *loop*. Então, para cada $i \in \{1, \ldots, n\}$, a i-ézima iteração do *loop* remove exatamente um ponto q de Q e produz o i-ézimo subconjunto, $X^{(i)}$, de uma sequência, $X^{(1)}, \ldots, X^{(n)}$, de subconjuntos de D, onde $X \subset X^{(1)}$ e $X^{(j-1)} \subset X^{(j)}$, para cada $j \in \{2, \ldots, n\}$. Mais especificamente, para cada $i \in \{1, \ldots, n\}$, o subconjunto

$X^{(i)}$ é obtido deixando $X^{(i)} = X^{(i-1)} \cup S^{(i)}$, onde $X^{(0)} = X$ e $S^{(i)} \subseteq (X_0 - X^{(i-1)})$. Em outras palavras, $S^{(i)}$ é um subconjunto do *background* da imagem $(D,\ X^{(i-1)})$, e portanto o *foreground*, $X^{(i)}$, de $(D,\ X^{(i)})$, é um superconjunto do *foreground*, $X^{(i-1)}$, de $(D,\ X^{(i-1)})$. Quando o *loop* termina, $X' = X^{(n)}$, o que é equivalente a dizer que $X' = X \cup P$, onde

$$P = \bigcup_{i=1}^{n} S^{(i)} \,. \tag{5.1}$$

Assim como o algoritmo proposto por Siqueira et al. [SLT+08], para computar o conjunto $S^{(i)}$, o algoritmo conta com três premissas simples, descritas abaixo.

1 Se q é o ponto removido de Q na i-ézima iteração e $\mathcal{I}(q)$ contém uma instância, A, de $C1$ e $C2$ em $(D,\ X^{(i-1)})$, então todos os pontos de A deve pertencer a D, uma vez que D é uma grade retangular.

2 Pode-se demonstrar que A não será uma configuração crítica em $(D,\ X^{(i)})$ somente se um subconjunto dos pontos do *background* de $(D,\ X^{(i-1)})$ que pertence a A está inserido em $S^{(i)}$. Então, o conjunto $S^{(i)}$ pode sempre ser computado e as escolhas dos pontos de $(X_0 - X^{(i-1)})$ a serem inseridos em $S^{(i)}$ estão limitados aos pontos do *background* de $(D,\ X^{(i-1)})$ em A.

3 O conjunto $\mathcal{I}(q)$ contém 0 (zero), ou exatamente um (ilustrado na Figura 5.2), ou exatamente duas (ilustrado na Figura 5.3), ou exatamente três (ilustrado na Figura 5.4) instâncias de $C1$ em $(D,\ X^{(i-1)})$, ou exatamente uma instância de $C2$ em $(D,\ X^{(i-1)})$ (possibilidades ilustradas nas Figuras 5.5 e 5.6). Ou seja, estes 5 (cinco) casos são mutuamente exclusivos.

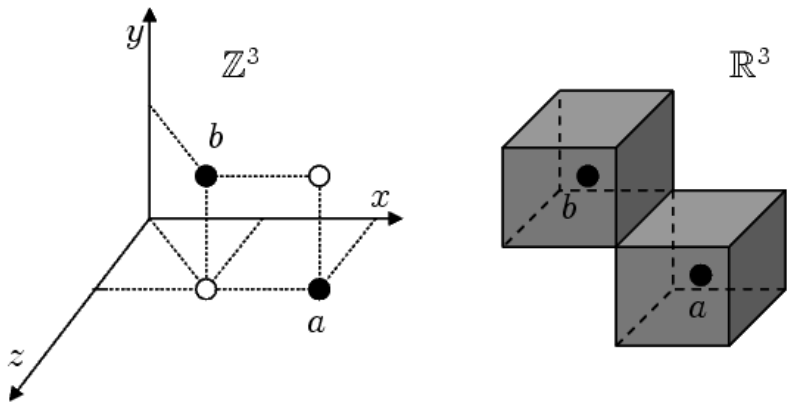

Figura 5.2: Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente uma única instância de $C1$. Os pontos a e b representam os dois pontos de *background* de $(D,\ X^{(i-1)})$ que pertencem a instância de $C1$. Figura extraída de Siqueira et al. [SLT+08].

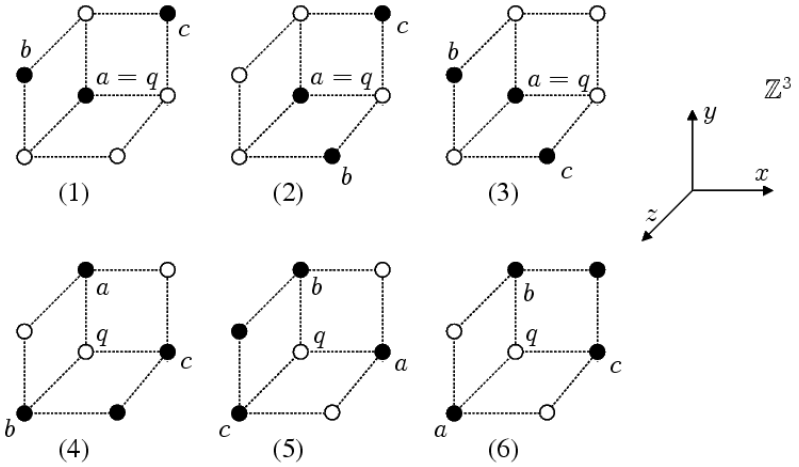

Figura 5.3: Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente duas instâncias de $C1$. Os pontos do *foreground* e do *background* de $(D, X^{(i-1)})$ que pertencem às duas instâncias de $C1$ estão representadas como círculos brancos e pretos, respectivamente. Figura extraída de Siqueira et al. [SLT+08].

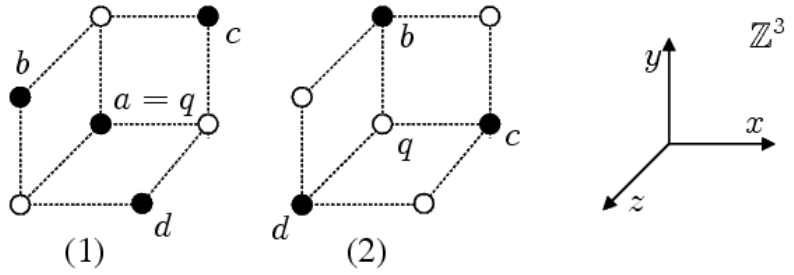

Figura 5.4: Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente três instâncias de $C1$. Os pontos do *foreground* e do *background* de $(D, X^{(i-1)})$ que pertencem às três instâncias de $C1$ estão representadas como círculos brancos e pretos, respectivamente. Figura extraída de Siqueira et al. [SLT+08].

Assim, o algoritmo computa um conjunto de conjuntos, denotado por $B(q)$, para cada um dos quatro casos mutuamente exclusivos nos quais $\mathcal{I}(q)$ contém uma instância de $C1$ e $C2$. Os elementos pertencentes a $B(q)$ são denominados de *escolhas* e $S(i)$ é selecionado a partir dos elementos pertencentes a $B(q)$. A Tabela 5.1 expõe as escolhas pertencentes a $B(q)$ em cada caso, e as Figuras 5.2 – 5.6 ilustram esses casos.

Para escolher um subconjunto $S^{(i)} \in B(q)$, o algoritmo segue a seguinte *regra da escolha*: selecione uma escolha, $S(i) \in B(q)$ tal que $(D, X^{(i)})$ não contém qualquer instância de $C1$ ou $C2$ que também não está em $(D, X^{(i-1)})$. Se existir mais de uma escolha, selecione aquele que apresentar menor grau de pertinência (casos A1, B(1) e B(2), ver Tabela 5.1) ou então, selecione aquele cuja soma dos graus de pertinência dos elementos apresentar o menor valor (casos A2, A3(1) e A3(2), ver Tabela 5.1). Se houver um empate, escolha um elemento de $B(q)$ randomicamente. Finalmente, se qualquer escolha é tal que $(D, X^{(i)})$ contém uma instância de $C1$ e $C2$ que não está em $(D, X^{(i-1)})$,

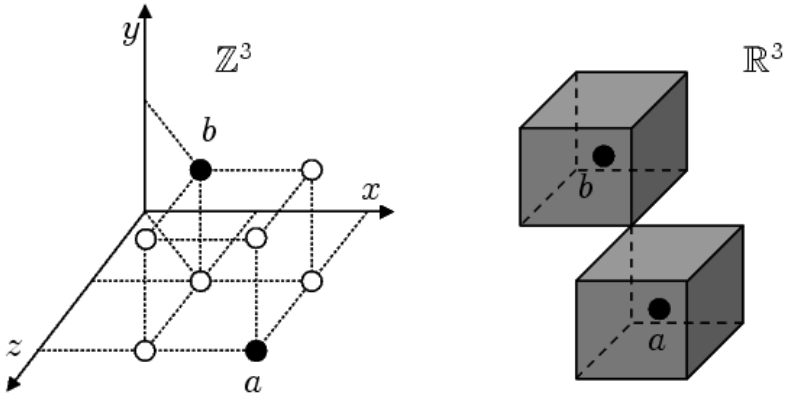

Figura 5.5: Ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente uma única instância de $C2$. Os pontos a e b representam os pontos de *background* de (D, $X^{(i-1)}$) que pertencem a instância de $C2$. Figura extraída de Siqueira et al. [SLT+08].

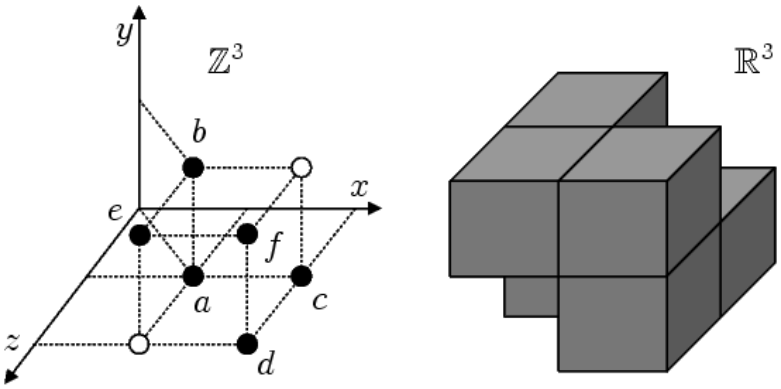

Figura 5.6: Outra ilustração do conjunto $\mathcal{I}(q)$ contendo exatamente uma única instância de $C2$. Os pontos a, b, c, d, e e f representam os pontos de *background* de (D, $X^{(i-1)}$) que pertencem a instância de $C2$. Figura extraída de Siqueira et al. [SLT+08].

siga a mesma regra descrita anteriormente: selecione aquele que apresentar menor grau de pertinência e, ocorrendo empate, selecione um randomicamente.

Para decidir se uma escolha para $S^{(i)}$ é tal que (D, $X^{(i)}$) contém uma instância de $C1$ ou $C2$ que não estava em (D, $X^{(i-1)}$), o algoritmo não precisa verificar todas as configurações $\mathcal{N}_x(p), \mathcal{N}_y(p), \mathcal{N}_z(p)$ e $\mathcal{N}(p)$, para todo $p \in D$. Desde que $X^{(i)} = X^{(i-1)} \cup S^{(i)}$, basta levar em consideração apenas os pontos que são 26-adjacentes a qualquer ponto em $\mathcal{N}(q)$, ou equivalentemente, os pontos da grade $4 \times 4 \times 4$

$$[q_1 - 1, q_1 + 2] \times [q_2 - 1, q_2 + 2] \times [q_3 - 1, q_3 + 2] \subset \mathbb{Z}^3, \tag{5.2}$$

onde (q_1, q_2, q_3) são as coordenadas de q. Desta forma, os testes para detectar novas configurações críticas em (D, $X^{(i)}$) podem ser feitos em tempo constante. Finalmente, para cada nova configuração crítica, $\mathcal{I}(p)$, em (D, $X^{(i)}$), o ponto p é inserido em Q.

Caso	C1	C2	B(q)	Figura
A1	1	0	$\{\{a\},\{b\}\}$	5.2
A2	2	0	$\{\{a\},\{b,c\}\}$	5.3
A3(1)	3	0	$\{\{a\},\{b,c,d\}\}$	5.4 (1)
A3(2)	3	0	$\{\{b,c\},\{b,d\},\{c,d\}\}$	5.4 (2)
B(1)	0	1	$\{\{a\},\{b\}\}$	5.5
B(2)	0	1	$\{\{a\},\{b\},\{c\},\{d\},\{e\},\{f\}\}$	5.6

Tabela 5.1: Descrição das escolhas pertencentes a $B(q)$ para cada um dos quatro casos mutuamente exclusivos (e suas subclasses) nos quais $\mathcal{I}(q)$ contém uma instância de C1 ou C2: denominação do caso (primeira coluna); número de instâncias de C1 em $\mathcal{I}(q)$ (segunda coluna); número de instâncias de C2 em $\mathcal{I}(q)$ (terceira coluna); as escolhas $\in B(q)$ (quarta coluna); e o número da figura que ilustra o referido caso (quinta coluna). Tabela extraída de Siqueira et al. [SLT+08].

5.2 Det3DWCFS - Um Algoritmo Reparador Determinístico para Imagens Bem-Compostas utilizando Segmentação *Fuzzy*

O algoritmo determinístico implementado segue a mesma lógica de raciocínio do algoritmo aleatório em relação ao fato de receber, como imagem de entrada (D, X), uma imagem segmentada pelo algoritmo MOFS [CHK05] e o respectivo grau de pertinência de cada ponto em relação ao objeto ao qual pertence; retornando uma imagem bem-composta (D, X') similar à de entrada. Sendo assim, para descobrir X', o algoritmo determinístico também encontra um subconjunto P do *background* X_0 de (D, X) tal que (D, $X \cup P$) é bem-composta, logo $X' = X \cup P$.

Da mesma forma que o algoritmo aleatório descrito na Seção 5.1, o algoritmo determinístico não garante encontrar o menor conjunto P, uma vez que ele busca encontrar o subconjunto P que apresenta a menor afinidade com o *background*. Contudo, ele apresenta complexidade de tempo polinomial em $\mid G \mid$.

No algoritmo determinístico, o processo de busca inicial pelas configurações críticas numa determinada imagem de entrada (D, X) também é similar ao do algoritmo aleatório, no qual pecorre-se todos os pontos de D procurando por instâncias de $C1$ e $C2$. Então, para cada ponto $p \in D$, analisa-se os pontos que são 26-adjacentes ao ponto p, conforme apresentado na Subseção 5.1.1. Entretanto, diferentemente do algoritmo aleatório, caso o ponto p esteja envolvido em alguma configuração crítica, testa se a sua mudança de cor, de 0 (*background*) para 1 (*foreground*), dará origem a novas configurações críticas. Em caso afirmativo, insere-o no *heap* mínimo $H2$. Do contrário, insere-o no *heap* mínimo $H1$.

No caso de nenhuma configuração crítica ser encontrada, ou seja, se os *heaps* $H1$ e $H2$ permanecerem vazios, então (D, X) não possui configurações críticas e o algoritmo termina com $X' = X$. Do contrário, o algoritmo parte para a etapa de eliminação das configurações críticas encontradas, conforme descrita na Subseção 5.2.1.

5.2.1 Eliminação das Configurações Críticas

No algoritmo determinístico, o passo da eliminação das configurações críticas resume-se como sendo um *loop* que é executado até que os *heaps* mínimos $H1$ e $H2$ tornem-se vazios. Esses *heaps* apresentam as seguintes propriedades:

- No *heap* $H1$, o nó raiz armazena o elemento (ponto $p \in X_0$) que apresenta o menor valor de grau de pertinência (calculado pelo algoritmo de segmentação *fuzzy*) e, caso haja empate, será aquele cuja mudança de objeto **eliminará** o maior número de configurações críticas.
- No *heap* $H2$, o nó raiz armazena o elemento (ponto $p \in X_0$) que apresenta o menor valor de grau de pertinência (calculado pelo algoritmo de segmentação *fuzzy*) e, caso haja empate, será aquele cuja mudança de objeto **gerará** o menor número de configurações críticas.

Enquanto $H1$ contiver algum elemento, o algoritmo se comporta da forma como descrito a seguir.

Sabemos que todos os elementos armazenados em $H1$ são pontos do *backgroud* envolvidos em configurações críticas, mas cuja mudança de cor não gera novas instâncias de $C1$ e $C2$. Logo, para cada elemento $h1$ retirado de $H1$, considera-se todos os pontos do *background* em uma vizinhança $3 \times 3 \times 3$, ou equivalentemente, os pontos da grade $3 \times 3 \times 3$

$$[h1_1 - 1, h1_1 + 2] \times [h1_2 - 1, h1_2 + 2] \times [h1_3 - 1, h1_3 + 2] \subset \mathbb{Z}^3,$$

onde $(h1_1, h1_2, h1_3)$ são as coordenadas de $h1$. Então, para cada ponto d pertencente ao *background* e vizinho ao ponto $h1$, o algoritmo verifica se d está presente em $H1$ e, posteriormente, em $H2$. Caso esteja contido em algum dos referidos *heaps*, analisa-se se ele ainda está envolvido em alguma configuração crítica. Caso não esteja, retira-o do *heap* em questão. Caso esteja, atualiza os *heaps* $H1$ e $H2$ com relação a d, caso necessário.

Posteriormente, o algoritmo analisará $H2$. Logo, enquanto este *heap* contiver algum elemento, o algoritmo se comporta da seguinte forma:

Sabemos que todos os elementos armazenados em $H2$ são pontos do *backgroud* envolvidos em configurações críticas e cuja mudança de cor gera novas instâncias de $C1$ e $C2$. Logo, para cada elemento $h2$ retirado de $H2$, considera-se todos os pontos do *background* em uma vizinhança $3 \times 3 \times 3$, ou equivalentemente, os pontos da grade $3 \times 3 \times 3$

$$[h2_1 - 1, h2_1 + 2] \times [h2_2 - 1, h2_2 + 2] \times [h2_3 - 1, h2_3 + 2] \subset \mathbb{Z}^3,$$

onde $(h2_1, h2_2, h2_3)$ são as coordenadas de $h2$. Então, para cada ponto m pertencente ao *background* e vizinho ao ponto $h2$, o algoritmo verifica se m está presente em $H1$ e, posteriormente, em $H2$. Caso esteja contido em algum dos referidos *heaps*, analisa-se se ele ainda está envolvido em alguma configuração crítica. Caso não esteja, retira-o do *heap* em questão. Caso esteja, atualiza os *heaps* $H1$ e $H2$ com relação a m, caso necessário.

5.2.2 Uma simplificação do Det3DWCFS

Visando obter uma investigação mais concisa com relação a eficácia do algoritmo determinístico Det3DWCFS, descrito anteriormente nesta mesma Seção, foi implementado o algoritmo determinístico Det3DWC. Este algoritmo é basicamente uma simplificação do Det3DWCFS, uma vez que a lógica de sua implementação consiste na mesma do Det3DWCFS, contudo sem utilizar informações sobre o grau de pertinência dos pontos do *background* durante a fase de eliminação das configurações críticas presentes na imagem de entrada.

Mais especificadamente, a única modificação que o Det3DWC sofreu em relação ao Det3DWCFS foram as propriedades dos *heaps* $H1$ e $H2$. As novas propriedades são descritas a seguir.

- No *heap* $H1$, o nó raiz armazena o elemento (ponto $p \in X_0$) cuja mudança de cor eliminará o maior número de configurações críticas.
- No *heap* $H2$, o nó raiz armazena o elemento (ponto $p \in X_0$) cuja mudança de cor gerará o menor número de configurações críticas.

Os demais passos seguem idênticos aos do algoritmo Det3DWCFS, conforme descrito na Subseção 5.2.1.

5.3 Experimentos

Nesta Seção são detalhados os procedimentos que foram realizados para as execuções dos experimentos com os 4 algoritmos descritos: o Ran3DWC, proposto por [SLT+08];

o Ran3DWCFS, descrito na Seção 5.1; o Det3DWCFS, apresentado na Seção 5.2; e o Det3DWC, comentado na Subseção 5.2.2. Além disso, são apresentados os parâmetros de comparação usados para avaliá-los e resultados obtidos.

Para avaliar o desempenho dos algoritmos, estes foram submetidos a testes utilizando imagens de ressonância magnética (MRI – do inglês, *Magnetic Resonance Image*) de dois tipos: imagens de ressonância magnéticas do cérebro humano captadas em T1, disponibilizadas pelo Prof. Dr. Alexandre Xavier Falcão, da Universidade Estadual de Campinas (Unicamp); e imagens simuladas de ressonância magnética do cérebro humano do Banco de Dados de Simulação de Cérebro (SBD – do inglês, *Simulated Brain Database*).

O SBD contém um conjunto de volumes de dados produzido pelo simulador de MRI 3D on-line, BrainWeb. Este banco de dados[1] permite que os usuários controlem vários parâmetros de aquisição, como por exemplo, a espessura da fatia (*slice*), tipo de pulso utilizado na aquisição (T1, T2 e PD) e nível do ruído, obtendo imagens de ressonância magnética realísticas do cérebro humano. A Figura 5.7 mostra fatias de um volume do cérebro, pertencente ao SBD. Este simulador utiliza os princípios de modelagem baseados na equação de Bloch [MW95] para implementar a simulação de um evento discreto de produção de sinais de ressonância magnética nuclear, modelos de ruídos realísticos e efeitos de volume parcial do processo de produção da imagem [CKK+97, CZK+98].

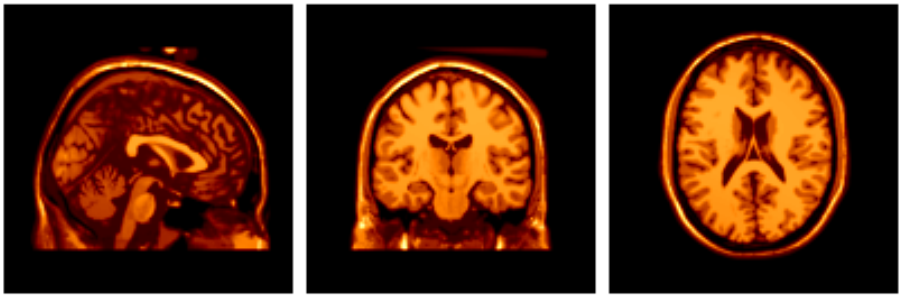

Figura 5.7: Fatias de um volume do cérebro humano simulado pelo BrainWeb. Visão do plano sagital, coronal e axial do volume, da esquerda para a direita.

Os experimentos dispuseram de seis volumes de MRI, obtidas do SBD, simulando um cérebro humano normal (i.e. sem anolamias). Estes volumes consistem numa mesma imagem de cérebro variando apenas o nível de ruído. A Tabela 5.2 apresenta esses volumes e seus respectivos parâmetros. Os testes também dispuseram de cinco volumes de imagens de ressonância magnética do cérebro humano captadas em T1, da Unicamp (aqui denominados de Brain 1, Brain 2, Brain 3, Brain 4 e Brain 5). A Figura 5.8 mostra imagens desses volumes.

Todos os volumes acima descritos foram inicialmente segmentados a partir do algo-

[1] Disponível em http://www.bic.mni.mcgill.ca/brainweb

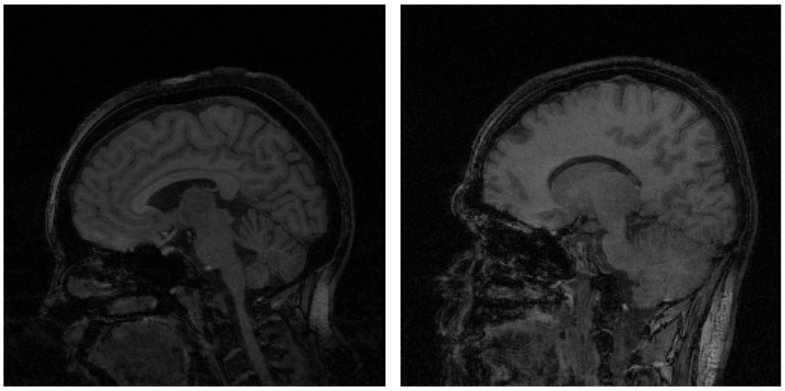

Figura 5.8: Fatias Sagitais de dois volumes distintos de MRI.

Volume	Modalidade	Espessura da Fatia	Nível de Ruído	Intensidade não-uniforme
BrainWeb 1	T1	1mm	0%	20%
BrainWeb 2	T1	1mm	1%	20%
BrainWeb 3	T1	1mm	3%	20%
BrainWeb 4	T1	1mm	5%	20%
BrainWeb 5	T1	1mm	7%	20%
BrainWeb 6	T1	1mm	9%	20%

Tabela 5.2: Apresentação dos volumes do cérebro simulados pelo BrainWeb e seus respectivos parâmetros.

ritmo MOFS, descrito na Subseção 4.3.1.1. Logo, para cada volume, foram escolhidos pontos sementes para cada objeto (parte do cérebro) a ser segmentado. Com o intuito de destacar a substância branca do restante do cérebro e do *background*, foi realizada uma 2-segmentação, segundo os passos descritos na Subseção 4.3.1.2. O volume de saída contém informações dos graus de pertinência de cada ponto em relação ao objeto ao qual pertence. A Figura 5.9 expõe o resultado da segmentação *fuzzy* de uma fatia axial dos volumes BrainWeb 1 e BrainWeb 6.

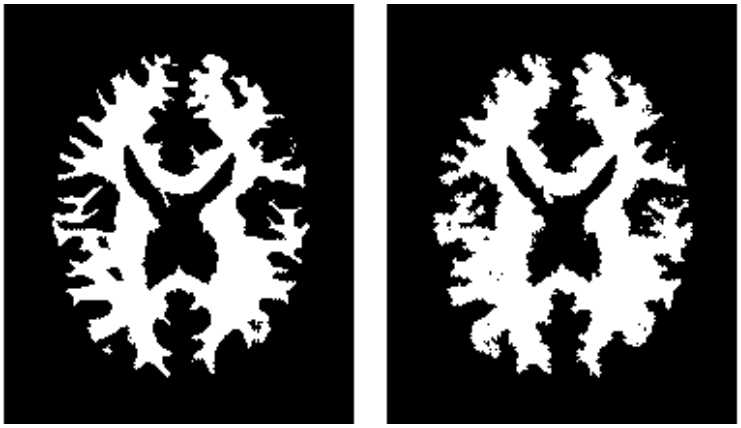

Figura 5.9: Imagens binárias resultantes da segmentação *fuzzy* dos volumes BrainWeb 1 (sem ruído, à esquerda) e BrainWeb 6 (com nível de ruído de 9%, à direita).

Posteriormente, executou-se cada um dos quatro algoritmos reparadores, acima men-

cionados, com cada um dos volumes segmentados, citados anteriormente. A Figura 5.10 apresenta os resultados das reparações de uma fatia axial do volume BrainWeb 1 executadas pelos algoritmos Ran3DWC e Ran3DWCFS. Por sua vez, a Figura 5.11 exibe imagens resultades da execução dos algoritmos Det3DWC e Det3DWCFS com mesmo volume testado (a saber, BrainWeb 1).

Figura 5.10: Imagem original da fatia axial do volume BrainWeb 1 (à esquerda), imagem resultante bem-composta gerada pelo Ran3DWC (ao centro), imagem resultante bem-composta gerada pelo Ran3DWCFS (à direita).

Figura 5.11: Imagem original da fatia axial do volume BrainWeb 1 (à esquerda), imagem resultante bem-composta gerada pelo Det3DWC (ao centro), imagem resultante bem-composta gerada pelo Det3DWCFS (à direita).

Tendo em vista que os algoritmos implementados buscam encontrar o subconjunto P que apresenta a menor afinidade com o *background* da imagem de entrada, conforme apresentado nas Seções 5.1 e 5.2, a métrica de similaridade que norteou os experimentos consistiu na razão da soma do total dos graus de pertinência pelo número de pontos modificados. Isto é justificado pelo fato desta métrica conter implicitamente uma medida que envolve o total de sigmas e o número de pontos.

Desta forma, para avaliar a eficácia dos algoritmos implementados, comparando-os com o algoritmo proposto por [SLT$^+$08], foi definidos parâmetros que pudessem auxiliar

a avaliação dos resultados obtidos dos mesmos. Assim, para cada execução, foram computados os seguintes parâmetros de comparação: quantidade de pontos do *background* modificados durante a transformação da imagem de entrada na sua bem-composta (chamado de N), somatório dos graus de pertinência (aqui denominado de *sigma*) dos pontos modificados e a divisão deste somatório dos graus de pertinência pela quantidade de pontos modificados (S/N).

Uma vez que os algoritmos Ran3DWC e Ran3DWCFS são aleatórios, cada um destes foram executados 10 (dez) vezes por imagem. Neste caso, também foram computados os desvios padrão para cada paramêtro de comparação.

Os resultados obtidos pela execução do algoritmo Ran3DWC tendo como imagem de entrada os volumes de imagens de ressonância magnética, Brain, são apresentadas na Tabela 5.3. A Tabela 5.4 mostra os valores obtidos com os volumes de imagens de ressonância magnética realísticas, BrainWeb.

Resultados Obtidos pelo Algoritmo Ran3DWC						
Volume	**Média do Total de Pontos Modificados**	**Desvio Padrão**	**Média do Total dos Sigmas**	**Desvio Padrão**	**Média do S/N**	**Desvio Padrão**
Brain 1	4887	17,71	2400,19	11,80	0,491188	0,001245
Brain 2	4764	14,79	1852,22	10,74	0,388792	0,001672
Brain 3	6545	18,03	2936,53	11,82	0,448646	0,001070
Brain 4	5217	15,19	2246,92	9,98	0,430692	0,001520
Brain 5	4705	9,51	2308,79	6,12	0,490763	0,001262

Tabela 5.3: Resultados obtidos pela execução do algoritmo Ran3DWC com os volumes de imagens de ressonância magnética Brain.

Resultados Obtidos pelo Algoritmo Ran3DWC						
Volume	**Média do Total de Pontos Modificados**	**Desvio Padrão**	**Média do Total dos Sigmas**	**Desvio Padrão**	**Média do S/N**	**Desvio Padrão**
BrainWeb 1	3620	8,01	1005,71	10,84	0,277842	0,002735
BrainWeb 2	4751	14,73	1587,21	10,33	0,334104	0,001429
BrainWeb 3	6569	16,06	2627,67	12,42	0,400040	0,001584
BrainWeb 4	7126	16,31	3247,29	20,49	0,455688	0,002483
BrainWeb 5	7496	21,31	3897,84	13,08	0,520004	0,001244
BrainWeb 6	7910	17,51	4643,33	8,89	0,587052	0,001476

Tabela 5.4: Resultados obtidos pela execução do algoritmo Ran3DWC com os volumes de imagens de ressonância magnética realística BrainWeb.

A Tabela 5.5 exibe os resultados obtidos pela execução do algoritmo Ran3DWCFS com os volumes de imagens de ressonância magnética, Brain. E na Tabela 5.6 são apresentados os valores obtidos com os volumes de imagens de ressonância magnética realísticas, BrainWeb.

Os resultados obtidos pelos experimentos realizados com o algoritmo determinístico Det3DWC a partir dos volumes de MRI Brain são apresentados na Tabela 5.7. E a Ta-

Resultados Obtidos pelo Algoritmo Ran3DWCFS						
Volume	**Média do Total de Pontos Modificados**	**Desvio Padrão**	**Média do Total dos Sigmas**	**Desvio Padrão**	**Média do S/N**	**Desvio Padrão**
Brain 1	4828	1,45	2033,27	0,98	0,421133	0,000109
Brain 2	4698	1,00	1484,78	0,55	0,316066	0,000066
Brain 3	6466	5,37	2457,94	2,75	0,380109	0,000146
Brain 4	5147	4,05	1835,79	1,01	0,356643	0,000209
Brain 5	4622	4,36	1949,58	2,06	0,421767	0,000157

Tabela 5.5: Resultados obtidos pela execução do algoritmo Ran3DWCFS com os volumes de imagens de ressonância magnética Brain.

Resultados Obtidos pelo Algoritmo Ran3DWCFS						
Volume	**Média do Total de Pontos Modificados**	**Desvio Padrão**	**Média do Total dos Sigmas**	**Desvio Padrão**	**Média do S/N**	**Desvio Padrão**
BrainWeb 1	3566	1,63	761,80	0,48	0,213659	0,000100
BrainWeb 2	4645	2,58	1172,26	1,41	0,252349	0,000145
BrainWeb 3	6411	2,27	2008,26	1,25	0,313243	0,000094
BrainWeb 4	7000	2,30	2608,24	1,24	0,372600	0,000096
BrainWeb 5	7359	3,20	3278,12	1,89	0,445475	0,000075
BrainWeb 6	7802	4,74	4073,15	3,69	0,522071	0,000168

Tabela 5.6: Resultados obtidos pela execução do algoritmo Ran3DWCFS com os volumes de imagens de ressonância magnética realística BrainWeb.

bela 5.8 mostra os valores obtidos com os volumes de imagens de ressonância magnética realísticas, BrainWeb.

Resultados Obtidos pelo Algoritmo Det3DWC			
Volume	**Total de Pontos Modificados**	**Total dos Sigmas**	**S/N**
Brain 1	4678	2272,14	0,485708
Brain 2	4606	1796,90	0,390121
Brain 3	6378	2850,75	0,446966
Brain 4	5034	2158,67	0,428818
Brain 5	4563	2245,00	0,492001

Tabela 5.7: Resultados obtidos pela execução do algoritmo Det3DWC com os volumes de imagens de ressonância magnética Brain.

Os resultados obtidos pela execução do algoritmo determinístico Det3DWCFS com os volumes de imagens de ressonância magnética, Brain, são apresentadas na Tabela 5.9. Na Tabela 5.10 são apresentados os valores obtidos com os volumes de imagens de ressonância magnética realísticas, BrainWeb.

5.4 Análise Comparativa dos Resultados

Os resultados obtidos durante os experimentos, apresentados na Seção 5.3, realizados com os 4 (quatro) algoritmos em questão serão analizados sob quatro perspectivas definidas a

Resultados Obtidos pelo Algoritmo Det3DWC			
Volume	**Total de Pontos Modificados**	**Total dos Sigmas**	**S/N**
BrainWeb 1	3508	958,31	0,273179
BrainWeb 2	4515	1477,38	0,327214
BrainWeb 3	6192	2440,97	0,394214
BrainWeb 4	6847	3107,03	0,453779
BrainWeb 5	7234	3762,60	0,520127
BrainWeb 6	7795	4586,74	0,588421

Tabela 5.8: Resultados obtidos pela execução do algoritmo Det3DWC com os volumes de imagens de ressonância magnética realística BrainWeb.

Resultados Obtidos pelo Algoritmo Det3DWCFS			
Volume	**Total de Pontos Modificados**	**Total dos Sigmas**	**S/N**
Brain 1	5091	2138,47	0,420049
Brain 2	4996	1580,40	0,316333
Brain 3	6929	2635,64	0,380378
Brain 4	5532	1991,79	0,360049
Brain 5	4956	2101,05	0,423940

Tabela 5.9: Resultados obtidos pela execução do algoritmo Det3DWCFS com os volumes de imagens de ressonância magnética Brain.

seguir.

1 VALOR DOS DESVIOS PADRÃO : Esta parte da análise investigará a variância dos resultados obtidos pelos algoritmos aleatórios Ran3DWC e o Ran3DWCFS quando executados dez vezes com cada volume de entrada.

2 QUANTIDADE DE PONTOS MODIFICADOS: Busca-se identificar qual(is) algoritmo(s) conseguiu(ram) gerar uma imagem binária 3D bem-composta similar a determinada imagem de entrada modificando um número menor de pontos do *background*.

Resultados Obtidos pelo Algoritmo Det3DWCFS			
Volume	**Total de Pontos Modificados**	**Total dos Sigmas**	**S/N**
BrainWeb 1	3667	780,19	0,212758
BrainWeb 2	4807	1208,87	0,251481
BrainWeb 3	6692	2081,86	0,311096
BrainWeb 4	7385	2731,87	0,369921
BrainWeb 5	7873	3503,29	0,444976
BrainWeb 6	8597	4521,58	0,525949

Tabela 5.10: Resultados obtidos pela execução do algoritmo Det3DWCFS com os volumes de imagens de ressonância magnética realística BrainWeb.

3 DIVISÃO DO TOTAL DOS GRAUS DE PERTINÊNCIA PELA QUANTIDADE DE PONTOS MODIFICADOS (N/S): Esta perspectiva de investigação visa apontar qual(is) algoritmo(s) conseguiu(ram) gerar uma imagem binária 3D bem-composta similar a determinada imagem de entrada modificando os pontos que apresentavam menor afinidade (grau de pertinência) em relação ao objeto ao qual fazia parte.

4 NÍVEL DE RUÍDO DAS IMAGENS: Esta investigará como os quatro algoritmos se comportam quando submetidos à imagens com níveis de ruídos distintos.

5.4.1 Análise em Relação ao Valor dos Desvios Padrão

Conforme citado anteriormente, os algoritmos Ran3DWC, proposto por [SLT+08], e Ran3DWCFS, por terem característica aleatória, foram executados 10 (dez) vezes para cada imagem das bases Brain e BrainWeb. Desta forma, também foram computados os desvios padrão para cada parâmetro de comparação.

Analisando os desvios padrão apresentandos nas Tabelas 5.3, 5.4, 5.5 e 5.6, podemos observar que, para estas bases, o algoritmo Ran3DWCFS apresentou uma diminuição considerável da dispersão dos valores obtidos quando submetido a uma mesma imagem (a saber, redução de até, aproximadamente, 93% de variação da quantidade de pontos modificados e redução de até 96% de variação da razão dos graus de pertinência pela quantidade de pontos modificados), em relação ao Ran3DWC. Isto pode ser explicado pelo fato do Ran3DWCFS tomar suas decisões baseando-se no grau de pertinência do mapa de conectividade *fuzzy* de cada ponto da imagem de entrada.

5.4.2 Análise em Relação à Quantidade de Pontos Modificados

Para facilitar a análise, as Figuras 5.12 e 5.13 apresentam o gráfico dos pontos modificados, por cada um dos quatro algoritmos considerados, nos volumes das bases Brain e BrainWeb, respectivamente.

Considerando os resultados obtidos para os algoritmos Ran3DWC e Ran3DWCFS, observamos que este apresentou redução de até aproximadamente 2,4%, quando submetidos aos volumes testados. Logo, concluímos que a adição de informações sobre o grau de pertinência dos mapas de conectividade *fuzzy* para auxiliar a tomada de decisão de quais pontos serão modificados, durante a reparação da imagem de entrada, contribuiu para a redução na quantidade total de pontos modificados.

Podemos observar também que o algoritmo Det3DWCFS apresentou em todos os casos analisados uma maior quantidade de pontos modificados. Isto se deve ao fato que

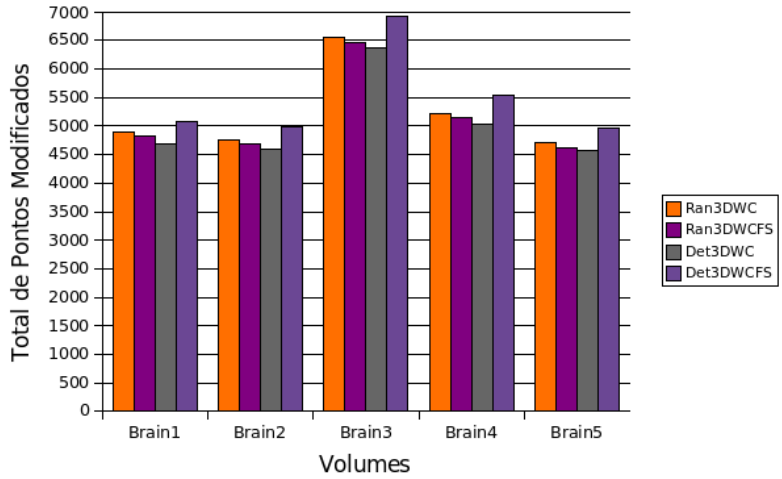

Figura 5.12: Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à quantidade de pontos modificados em cada volume da base Brain.

durante a sua implementação visou-se que a referida tomada de decisão se baseasse *principalmente* na seleção de pontos que apresentassem menor grau de pertinência e, apenas nos casos de haver empate entre pontos é que leva-se em conta a seleção daquele cuja mudança de objeto reduziria a quantidade de novas configurações críticas geradas, conforme descrito na Seção 5.2.1.

Por outro lado, o algoritmo Det3DWC foi o que apresentou a menor quantidade de pontos modificados. Isto é explicado pelo fato que sua tomada de decisão leva-se em conta apenas a seleção daqueles pontos cuja mudança de objeto reduziria a quantidade de novas configurações críticas geradas, conforme descrito na Subseção 5.2.2.

5.4.3 Análise em Relação à Divisão do Total dos Graus de Pertinência pela Quantidade de Pontos Modificados

As Figuras 5.14 e 5.15 apresentam o gráfico dos pontos modificados, por cada um dos quatro algoritmos considerados, nos volumes das bases Brain e BrainWeb, respectivamente.

Os resultados apresentados na referida figura mostram que os algoritmos Det3DWC e Det3DWCFS apresentaram, em todos os volumes testados, uma redução considerável no valor do grau de pertinência por ponto modificado.

Baseando-se no conceito *fuzzy*, temos que aqueles pontos que apresentarem o menor grau de pertinência (indicando a menor certeza dele pertencer a determinado objeto) estarão mais *aptos* a sofrerem mudança de objeto. Portanto, acredita-se que, a redução do

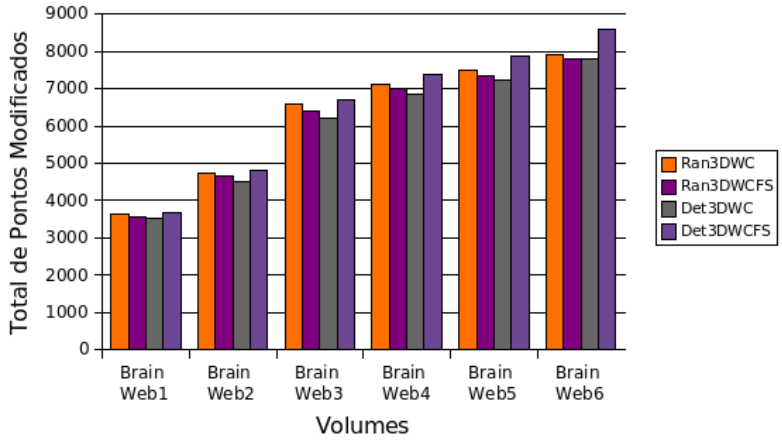

Figura 5.13: Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à quantidade de pontos modificados em cada volume da base BrainWeb.

somatório dos graus de pertinência dos pontos modificados numa imagem a ser reparada, independente da quantidade de pontos modificados, conduz a uma melhora na imagem resultante, no que diz respeito a sua similaridade topológica em relação ao objeto real.

5.4.4 Análise em Relação ao Nível de Ruído das Imagens

Conforme citado anteriormente, os volumes da base BrainWeb consistem numa mesma imagem de cérebro variando apenas o nível de ruído. No caso dos volumes com ruído, a vantagem de se utilizar os algoritmos que usam os mapas de conectividade fuzzy diminui a medida que o nível de ruído aumenta.

Assim, avaliando as Tabelas 5.4, 5.6, 5.8 e 5.10, podemos observar que, para esta base, os quatro algoritmos obtiveram aumento considerável dos resultados obtidos, tomando por base o volume sem ruído, BrainWeb1. Concluimos, então, que em presença de ruído aumenta-se a quantidade de pontos a serem modificados a fim de tornar uma determinada imagem na sua bem-composta.

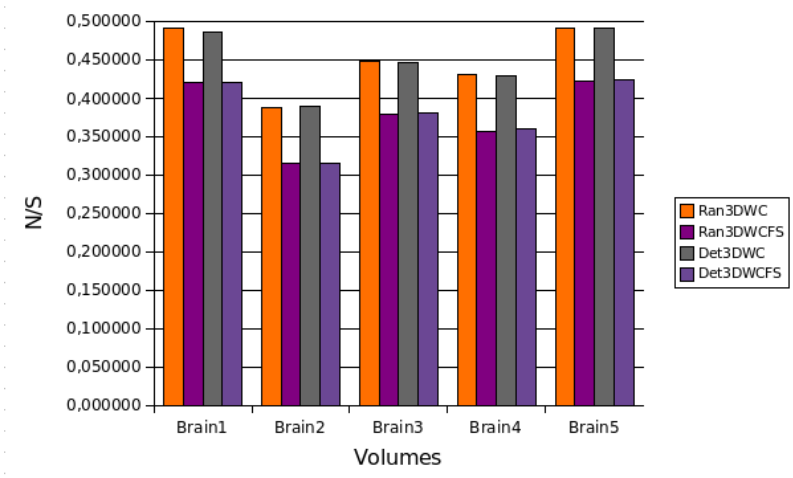

Figura 5.14: Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à divisão do total dos graus de pertinência pela quantidade de pontos modificados (N/S) em cada volume da base Brain.

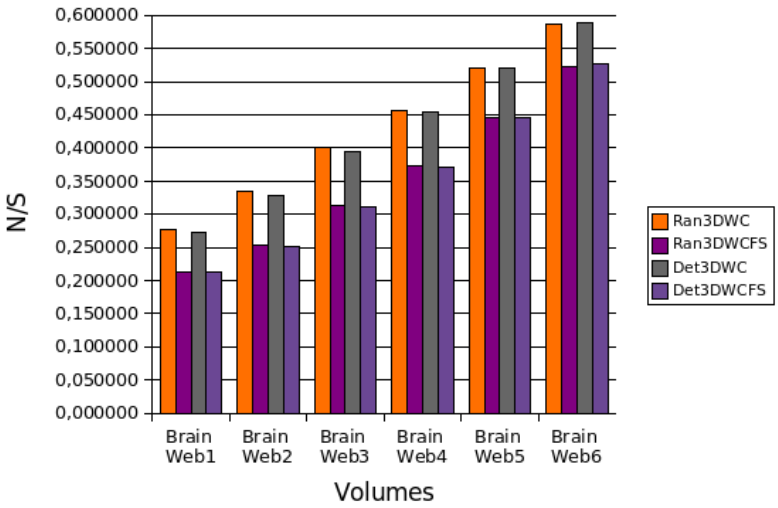

Figura 5.15: Comparação dos algoritmos Ran3DWC, Ran3DWCFS, Det3DWC e Det3DWCFS em relação à divisão do total dos graus de pertinência pela quantidade de pontos modificados (N/S) em cada volume da base BrainWeb.

Capítulo 6

Conclusão

Como vimos, imagens bem-compostas se favorecem de interessantes propriedades topológicas que nos permitem simplificar e otimizar a velocidade de algoritmos comumente usados na computação gráfica, processamento de imagens e visão computacional. Mesmo sem haver garantias que a imagem resultante será topologicamente igual ao do objeto real, estas vantagens tem motivado o desenvolvimento de algoritmos para reparar imagens bi e tridimensionais que não são bem-compostas, os chamados algoritmos reparadores.

Este trabalho propôs a utilização de informações resultantes do próprio processo de segmentação *fuzzy* de imagens, durante a reparação de uma imagem digital binária 3D na sua bem-compostura. Como forma de analisar a referida proposta, foi desenvolvido um algoritmo reparador aleatório capaz de realizar reparos topológicos em imagens binárias 3D a fim de produzir imagens bem-compostas similares às imagens dadas. Para tanto, este algoritmo modifica o valor atribuído de alguns pontos da imagem de entrada de 0 (*background*) para 1 (*foreground*), baseando-se no mapa de conectividade *fuzzy*, resultante da segmentação *fuzzy* realizada na própria imagem num passo anterior. Este algoritmo foi comparado com outro similar encontrado na literatura, porém que não usa qualquer informação para esta tomada de decisão.

Também foi desenvolvido um algoritmo determinísto que, para realizar a reparação das imagens digitais binárias 3D, baseia-se na quantidade de configurações críticas que venham a surgir com essa mudança e no grau de pertinência de cada ponto do *background*. Visando obter uma investigação mais concisa com relação a eficácia deste algoritmo foi implementado um algoritmo determinístico similar, contudo sem utilizar informações sobre o grau de pertinência dos pontos do *background* durante a fase de eliminação das configurações críticas presentes na imagem de entrada.

O processo de investigação consistiu em comparar os algoritmo acima citados. Para tanto, os mesmo foram submetidos a imagens de ressonância magnética (com e sem ruí-

dos) segmentados pelo algoritmo MOFS, proposto por [CHK05]. As conclusões tiradas a partir dos experimentos serão apontadas a seguir.

A utilização do grau de pertinência de cada *voxel*, no processo de decisão de quais deverão ser modificados durante a reparação de uma imagem 3D, resultou na eliminação da aleatoriedade do algoritmo proposto por Siqueira et al. [SLT+08], além da redução da quantidade total de pontos modificados por este.

O algoritmo determinístico implementado utilizando informação resultante do mapa de conectividade *fuzzy* apresentou, em todos os testes, o menor valor da divisão do total dos graus de pertinência pela quantidade de pontos modificados. Baseando-se no conceito *fuzzy*, temos que aqueles pontos que apresentarem o menor grau de pertinência (indicando a menor certeza dele pertencer a determinado objeto) estarão mais *aptos* a sofrerem mudança de objeto. Assim, acredita-se que a adição da informação dos graus de pertinência dos pontos, resultante de uma segmentação *fuzzy*, conduz a uma melhora na imagem resultante, no que diz respeito a sua similaridade topológica em relação ao objeto real.

6.1 Trabalhos Futuros

O estudo realizado neste trabalho ainda pode ser expandido. Dessa forma, alguns trabalhos futuros possíveis são descritos a seguir.

- Realizar mais testes, inclusive com outros tipos de imagens, como por exemplo, Tomografia Computadorizada (CT), Tomografia de Emissão de Pósitron (PET), Tomografia Computada por Emissão de Fóton Único (SPECT).
- Analisar a similaridade topológica das imagens bem-compostas reparadas pelo algoritmo determinístico apresentado no presente trabalho, com seus respectivos objetos reais.
- Extensão dos algoritmos para tratar imagens não-binárias (i.e. com mais de dois objetos).

Referências Bibliográficas

[BK03] S. Bischoff and L. Kobbelt. Sub-voxel topology control for level-set surfaces. *Computer Graphics Forum*, 22(3):273–280, 2003.

[BP05] P. L. Bazin and D. L. Pham. Topology correction using fast marching methods and its application to brain segmentation. In *Proceedings of the 8th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 3750 of *Lecture Notes in Computer Science*, pages 484–491, Palm Springs, CA, EUA, 2005. Springer.

[CGHK99] B. M. Carvalho, C. J. Gau, G. T. Herman, and T. Yung Kong. Algorithms for fuzzy segmentation. *Pattern Analysis and Applications*, 2(1):73–81, 1999.

[CHK05] B. M. Carvalho, G. T. Herman, and T. Yung Kong. Simultaneous fuzzy segmentation of multiple objects. *Discrete Applied Mathematics*, 105(1-3):55–77, 2005.

[CKK+97] C.A. Cocosco, V. Kollokian, R. K. S. Kwan, G. B. Pike, J. Sled, and A. Evans. Brainweb: Online interface to a 3D MRI simulated brain database. In *Proceedings of 3-rd International Conference on Functional Mapping of the Human Brain*, volume 5, page 425, Copenhagen, 1997. NeuroImage.

[COG6a] B. M. Carvalho, L. M. Oliveira, and E. Garduno. Semi-automatic single particle segmentation on electron micrographs: From nano to macro. In *International Symposium on Biomedical Imaging*, pages 1024–1027, Washington, DC, USA, 2006a. IEEE.

[CON6b] B. M. Carvalho, L. M. Oliveira, and L. S. Britto Neto. Bottled sand movies. In *CGIV 0́6: Third International Conference on Computer Graphics, Imaging and Visualisation*, pages 402–407, Sydney, Austrália, 2006b. IEEE.

[CZK+98] D. Collins, A. Zijdenbos, V. Kollokian, J. Sled, N. Kabani, C. Holmes, and A. Evans. Design and construction of a realistic digital brain phantom. *IEEE Transactions on Medical Imaging*, 17(3):463–468, 1998.

[EL94] U. Eckhardt and L. J. Latecki. *Digital Topology*. Research Trends, Vilayil Gardens, Trivandrum, India, 1994.

[Fac93] J. Facon. *Processamento e análise de imagens*. VI EBAI, Córdoba, 1993.

[FLD01] B. Fischl, A. Liu, and A. M. Dale. Automated manifold surgery: Constructing geometrically accurate and topologically correct models of the humam cerebral cortex. *IEEE Transactions on Medical Imaging*, 20(1):70–80, 2001.

[GW02] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, EUA, 2 edition, 2002.

[Har69] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.

[HC01] G. T. Herman and B. M. Carvalho. Multiseeded segmentation using fuzzy conectedness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):460–474, 2001.

[Her98] G. T. Herman. *Geometry of digital spaces*. Birkhauser, Boston, MA, USA, 1998.

[HXBNP02] X. Han, C. Xu, U. Braga-Neto, and J. L. Prince. Topology correction in brain cortex segmentation using a multiscale, graph-based approach. *IEEE Transactions on Medical Imaging*, 21(2):109–121, 2002.

[HXP03] X. Han, C. Xu, and J. L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):755–768, 2003.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computer Surveys*, 31(3):264–323, 1999.

[KG01] N. Kriegeskorte and R. Goeble. An efficient algorithm for topologically segmentation of the cortical sheet in anatomical MR volumes. *Neuroimage*, 14(2):329–346, 2001.

[Lat97] L. J. Latecki. 3D well-composed pictures. *Graphical Models and Image Processing*, 59(3):164–172, 1997.

[Lat98] L. J. Latecki. *Discrete Representation of Spatial Objects in Computer Vision*. Kluwer Academic Publishers, Norwell, MA, USA, 1 edition, 1998.

[LCG98] L. J. Latecki, C. Conrad, and A. Gross. Preserving topology by a digitalization process. *Journal of Mathematical Imaging and Vision*, 8(2):131–159, 1998.

[LER95] L. J. Latecki, U. Eckhardt, and A. Rosenfeld. Well-composed sets. *Computer Vision and Image Understanding*, 61(1):70–83, 1995.

[LL99] K. C. Laudon and J. P. Laudon. *Sistemas de informação*. LTC, Rio de Janeiro, 4 edition, 1999.

[MFB+95] J. F. Mangin, V. Frouin, I. Bloch, J. Régis, and J. López-Krahe. From 3D magnetic resonance images to structural representations of the cortex topography using topology preserving deformations. *Journal of Mathematical Imaging and Vision*, 5(4):297–318, 1995.

[MKAE00] D. MacDonald, N. Kabsni, D. Avis, and A. C. Evans. Automated 3D extraction of inner and outer surfaces of cerebral cortex from mri. *Neuroimage*, 12(3):340–355, 2000.

[MV89] N. D. D. Mascarenhas and F. R. D. Velasco. *Processamento Digital de Imagens*. Kapelusz, Buenos Aires, 2 edition, 1989.

[MW95] L. Mandel and E. Wolf. *Optical Coherence and Quantum Optics*. Cambridge University Press, Cambridge, 1995.

[PM86] S. K. Pal and D. K. D. Majumder. *Fuzzy mathematical approach to pattern recognition*. Halsted Press, New York, NY, USA, 1986.

[RKN98] A. Rosenfeld, T. Yung Kong, and A. Nakamura. Topology-preserving deformations of two-valued digital pictures. *Graphical Models and Image Processing*, 60(1):24–34, 1998.

[Ros79] A. Rosenfeld. Fuzzy digital topology. *Information and Control*, 40(1):76–87, 1979.

[SGF05] F. Ségonne, W. E. L. Grimson, and B. Fischl. A genetic algorithm for the topology correction of cortical surfaces. In *International Conference on Information Processing in Medical Imaging*, volume 3565 of *Lecture Notes in Computer Science*, pages 393–405, Colorado, USA, 2005. Glenwood Springs.

[SK05] P. Stelldinger and U. Köthe. Towards a general sampling theory for shape preservation. *Image and Vision Computing Journal*, 23(2):237–248, 2005.

[SL01] D. W. Shattuck and R. M. Leahy. Automated graph-based analysis and correction of cortical volume topology. *IEEE Transactions on Medical Imaging*, 20(11):464–472, 2001.

[SLG04] M. Siqueira, L. J. Latecki, and J. Gallier. Making 3d binary digital images well-composed. Technical report, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA, 2004.

[SLS07] P. Stelldinger, L. J. Latecki, and M. Siqueira. Topological equivalence between a 3D object and the reconstruction of its digital image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):126–140, 2007.

[SLT+08] M. Siqueira, L. J. Latecki, N. Tustison, J. Gallier, and J. Gee. Topological repairing of 3D digital images. *Journal of Mathematical Imaging and Vision*, 30(3):249–274, 2008.

[US96] J. K. Udupa and S. Samarasekera. Fuzzy connectedness and object definition: Theory, algorithms and applications in image segmentation. *Graphical Models and Image Processing*, 58(3):246–261, 1996.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas