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
MESTRADO ACADÊMICO EM SISTEMAS E COMPUTAÇÃO

SALSA – Simple Automatic Lung Segmentation Algorithm

Addson Araújo da Costa

Natal-RN
Junho de 2016

Addson Araújo da Costa

SALSA – Simple Automatic Lung Segmentation Algorithm

Dissertação de Mestrado apresentada 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 requisito parcial para a obtenção do grau de Mestre em Sistemas e Computação.

Linha de pesquisa:

Processamento de Imagens

Orientador

Doc. Bruno Motta de Carvalho

PPGSC – PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO
DIMAP – DEPARTAMENTO DE INFORMÁTICA E MATEMÁTICA APLICADA
CCET – CENTRO DE CIÊNCIAS EXATAS E DA TERRA
UFRN – UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Natal-RN

Junho de 2016

Catálogo da Publicação na Fonte. UFRN / SISBI / Biblioteca Setorial
Especializada do Centro de Ciências Exatas e da Terra – CCET.

Costa, Addson Araújo da.

SALSA – Simple Automatic Lung Segmentation Algorithm / Addson Araújo da Costa. – Natal, RN, 2016.

72 f. : il.

Orientador: Prof. Dr. Bruno Motta de Carvalho.

Dissertação (Mestrado) – 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.

1. Processamento de imagens – Dissertação. 2. Thresholding – Dissertação. 3. Detecção de borda – Dissertação. 4. Algoritmo FloodFill – Dissertação. 5. Algoritmo de Watershed – Dissertação. 6. Segmentação de imagens – Dissertação. I. Carvalho, Bruno Motta de. II. Título.

RN/UF/BSE-CCET

CDU 004.932

ADDSON ARAÚJO DA COSTA

SALSA – A Simple Automatic Lung Segmentation Algorithm

Esta Dissertação foi julgada adequada para a obtenção do título de mestre em Sistemas e Computação e aprovado em sua forma final pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Dr. Bruno Motta de Carvalho – UFRN
(Presidente)

Dr. Uirá Kulesza – UFRN
(Coordenador do Programa)

Banca Examinadora

Dr. Bruno Motta de Carvalho – UFRN
(Examinador)

Dr. Arne Magaly de Paula Canuto – UFRN
(Examinador)

Dr. Rafael Beserra Gomes – UFRN
(Examinador)

Dr. Wilfredo Blanco Figuerola – UERN
(Examinador)

Julho, 2016

Dedico esse trabalho aos meus amigos e familiares por toda o incentivo que me foi dado.

Agradecimentos

Agradeço ao meu professor orientador pela paciência e atenção dada ao longo do projeto.

SALSA – A Simple Automatic Lung Segmentation Algorithm

Autor: Addson Araújo da Costa

Orientador(a):

Doc. Bruno Motta de Carvalho

RESUMO

A segmentação precisa de tecidos pulmonares é de grande importância em várias tarefas de diagnóstico. Um simples e rápido algoritmo para segmentação de tecidos pulmonares é proposto aqui. O método combina várias operações simples de processamento de imagem para alcançar a segmentação final e pode ser dividido em dois problemas. O primeiro é a segmentação pulmonar, que consiste na identificação de regiões como fundo, traqueia, veias, pulmões esquerdo e direito, que é um processo dificultado pela presença de ruído, artefatos, baixo contraste, doenças e remoções cirúrgicas. O segundo é a segmentação dos lóbulos pulmonares, onde o pulmão esquerdo deve ser dividido em dois lóbulos, superior e inferior, e o pulmão direito deve ser dividido em três lóbulos, superior, inferior e médio. Essa segundo problema é dificultado devido ao fato de que a membrana que separa os lóbulos, a pleura visceral, é muito delicada e não é claramente visível nos exames de tomografia computadorizada, além da ocorrência de procedimentos de lobectomia (remoção cirúrgica de lóbulos), doenças que degradam a qualidade da imagem, ou ruído na aquisição da imagem. Ambos os métodos foram desenvolvidos de forma a tornar todo o procedimento automático, tendo resultados similares a outros trabalhos já publicados na área de segmentação pulmonar, utilizando a metodologia de teste do desafio de segmentação de pulmões LOLA11.

Palavras-chave: Thresholding, Detecção de Borda, Algoritmo FloodFill, Algoritmo de Watershed, Segmentação de Imagens.

SALSA – A Simple Automatic Lung Segmentation Algorithm

Author: Addson Araújo da Costa

Supervisor: Doc. Bruno Motta de Carvalho

ABSTRACT

The accurate segmentation of pulmonary tissue is of great importance for several diagnostic tasks. A simple and fast algorithm for performing lung segmentation is proposed here. The method combines several simple image processing operations to achieve the final segmentation and can be divided into two problems. The first is the lung segmentation, that identifies regions such as background, trachea, vessels, and left and right lungs, and it is complicated by the presence of noise, artifacts, low contrast and diseases. The second is the lobe segmentation, where the left lung is divided into two lobes, the upper and lower lobes, and the right into three lobes, the upper, middle and lower lobes. This second problem is harder due to the fact that the membranes dividing the lobes, the pleurae, are very thin and are not clearly visualized in the computerized tomography exams, besides the possible occurrence of lobectomies (surgical lobe removal), diseases that may degrade the image quality, or noise during image acquisition. Both methods were developed in order to produce an automatic method, and we have already produced results similar to the ones produced by other studies for the first problem, using the testing methodology of the lung segmentation challenge LOLA11.

Keywords: Thresholding, Boundary Detection, FloodFill Algorithm, Watershed Algorithm, Image Segmentation.

Lista de figuras

1.1	Imagens capturadas utilizando TC. (a) Visão tridimensional, (b) Fatia Axial, (c) Fatia Coronal e (d) Fatia Sagital.	p. 14
1.2	Vizinhança (a) de aresta, (b) de vértice e (c) face, (d) completa de um voxel.	p. 15
1.3	Casos problema encontrados no banco de dados de LOLA11. (a) Má formação do pulmão, (b) Pulmão com nódulos, (c) Artefatos e (d) Alto Ruído.	p. 16
1.4	Divisão de lóbulos de acordo com as fissuras. Fonte: (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014)	p. 18
2.1	Espaço de escala gaussiana $L(x, y, t)$ com tamanho de kernel variando entre zero e vinte e cinco. (a) Tamanho zero, correspondendo a imagem original, (b) cinco, (c) quinze e (d) vinte e cinco.	p. 21
2.2	Exemplo de MRF modelado usando grafo. As arestas indicam dependência entre as variáveis que são modeladas como nós.	p. 22
2.3	Fluxograma descrevendo algoritmo. Fonte: Lassen et al. (LASSEN et al., 2011).	p. 26
2.4	Brilho de elementos no entorno e os planos cortando um voxel utilizados na detecção da fissura. Fonte: Lassen et al. (LASSEN et al., 2011).	p. 27
3.1	Fluxo geral do SALSA. (a) Imagem Original, (b) Pré-processamento da imagem, (c) Filtro de Sobel, (d) Detecção da borda, (e) Remoção do background, (f) Segmentação da traqueia, (g) Corte de Pulmões, (h) Segmentação de vasos, (i) Segmentação final.	p. 29
3.2	Imagem Axial antes e depois da aplicação do Threshold.	p. 34
3.3	Sementes detectadas. (a) Imagem obtida após o processo de <i>thresholding</i> e (b) Sementes identificadas.	p. 34

3.4	Automato Finito usado para detectar sementes na borda pulmonar, Fluxograma geral do algoritmo.	p. 35
3.5	Aplicação de filtros sobre imagem. (a) Imagem axial com pouco ruído Lola11-05, (b) Imagem axial com muito ruído Lola11-07, (c) Sobel sobre Lola11-05 e (d) Sobel sobre Lola11-07.	p. 39
3.6	Deteccção de borda. (a) Visão Tridimensional e (b) Visão Axial.	p. 42
3.7	Segmentação de <i>background</i> . (a) Antes e (b) após segmentar o <i>background</i> em duas etapas.	p. 43
3.8	Segmentação da Traqueia.	p. 45
3.9	Divisão Pulmonar. (a) Corte inicial, (b) Complementação e (c) Otimização. Imagem meramente ilustrativa, pois com o auxílio do filtro de Sobel usado na detecção de bordas, esse método tem impacto menor sobre a imagem ilustrada considerando que o contorno é melhor detectado. . .	p. 48
3.10	Segmentação de Vasos Sanguíneos. (a) antes e (b) depois de verificar vizinhança dos voxels usando algoritmo <i>k-NN</i>	p. 50
3.11	Resultado final. (a) Axial (b) Tridimensional.	p. 52
4.1	Objetivo da segmentação de veias, subárvores de veias formadas por lóbulo.	p. 53
4.2	Imagens geradas pelos valores de lambda. Usando lambda 1, os planos de corte (a) axial, (b) sagital e (c) coronal, usando lambda 2, os planos de corte (d) axial, (e) sagital e (f) coronal e com lambda 3, os planos de corte (g) axial, (h) sagital e (i) coronal.	p. 58
4.3	Segmentação de veias, visão de cima. (a) Veias obtidas com o algoritmo obtido na Seção 3.6 (b) Veias obtidas usando Autovalores λ_2 sobre a matriz hessiana e (c) Subtração das estruturas 3D que geraram as imagens anteriores	p. 60
4.4	Mapeamento de distâncias, com e sem uso de erosão sobre as veias antes de mapear a distância. (a) e (c) Pulmão esquerdo, (b) e (d) Pulmão direito, respectivamente sem erosão e com uso de erosão antes de mapear as distâncias. Em destaque as duas maiores regiões conexas de veias para o pulmão esquerdo e as três maiores regiões para o pulmão direito. . . .	p. 61

4.5	Segmentação de veias baseado em sementes. Lóbulo superior em vermelho e bege, respectivamente, do pulmão direito e esquerdo. Lóbulo inferior em verde e marrom, respectivamente, pertencentes ao pulmão direito e esquerdo. Em azul o lóbulo médio e em cinza, veias não segmentadas. .	p. 62
4.6	Estágios do algoritmo de watershed (a) imagem original, (b) visão topográfica com pontos semente escolhidos e (c) inundação inicial, (d) inundação completa. Fonte: Digital Image Processing (GONZALEZ; WOODS, 2002)	p. 63
4.7	Imagem de custo e resultado final da segmentação obtido pela aplicação do algoritmo <i>watershed</i> sobre a matriz de custo obtida combinando a imagem gerada por λ_3 e o mapeamento de distância entre veias.	p. 64
5.1	Segmentação de pulmões SALSA, a esquerda, o padrão de referência e a direita, o resultado obtido pelo SALSA. (a-b) Lola11-25, (c-d) Lola11-37 e (e-f) Lola11-45	p. 66
5.2	Segmentação de lóbulos pulmonares. (a) Lola11-01, (b) Lola11-02, (c) Lola11-03, (d) Lola11-05	p. 69

Lista de Algoritmos

1	Algoritmo para criação do modelo adaptativo baseado em aparência (SOLIMAN et al., 2015)	p. 23
2	Algoritmo para segmentação de pulmões (SOLIMAN et al., 2015)	p. 23
3	FloodFill2D com vizinhança de aresta	p. 28
4	Ler arquivo de imagem	p. 32
5	Thresholding da imagem	p. 33
6	Busca por sementes	p. 35
7	Estimação de nível de ruído	p. 40
8	Filtro de Sobel	p. 40
9	Algoritmo de Busca por Borda	p. 41
10	Algoritmo para Remoção do Fundo	p. 43
11	Segmentação da traqueia	p. 45
12	Divisão Pulmonar	p. 48
13	Separação de veias	p. 51
14	Algoritmo Watershed de Meyer	p. 64

Sumário

1	Introdução	p. 13
1.1	Objetivo	p. 17
1.2	Organização do trabalho	p. 18
2	Trabalhos Relacionados	p. 19
2.1	Segmentação de pulmões	p. 20
2.2	Segmentação de lóbulos pulmonares	p. 25
3	SALSA – Simple Automatic Lung Segmentation Algorithm	p. 28
3.1	Pré-processamento da imagem	p. 32
3.1.1	Thresholding	p. 33
3.1.2	Detecção de sementes	p. 34
3.1.3	Filtragem espacial	p. 37
3.2	Busca por borda	p. 41
3.3	Remoção de background	p. 42
3.4	Segmentação de traqueia	p. 44
3.5	Divisão pulmonar	p. 47
3.6	Vasos sanguíneos	p. 50
3.7	Segmentação final	p. 52
4	Segmentação de Lóbulos Usando o SALSA	p. 53
4.1	Complementação de veias	p. 54
4.2	Segmentação de veias em lóbulos	p. 59

4.3	Watershed	p. 62
5	Discussão e Resultados	p. 65
5.1	Resultados da segmentação pulmonar	p. 65
5.2	Resultados da segmentação de lóbulos pulmonares	p. 68
6	Considerações Finais	p. 70
	Referências	p. 71

1 Introdução

O Diagnóstico Auxiliado por Computador (do Inglês, *CAD*) é uma área em grande expansão devido à necessidade de diagnóstico rápido e preciso sobre grandes e complexas imagens. Na pneumologia ¹, a segmentação dos pulmões é o primeiro passo para o diagnóstico, mas é uma tarefa de alto consumo de tempo caso seja executada de forma manual devido ao grande volume de dados, degradações causadas por doenças, ruído, complexidade de visualização dos dados por ser uma imagem tridimensional, entre outros fatores (DOEL; GAVAGHAN; GRAU, 2014).

Para gerar as imagens apresentadas neste trabalho, foram usados as ferramentas ITK-Snap (ITK-SNAP, Acessado em Novembro, 2014) para gerar imagens com resultados de segmentação, imagens tomográficas axiais, sagitais e coronais, além disso foi usado a ferramenta de edição gráfica GIMP (GIMP, Acessado em Maio, 2015) para recortes e gerar resultados de filtragens espaciais e foi usado também o SketchUp (SKETCHUP, Acessado em Maio, 2015) para gerar modelos tridimensionais de vizinhança. As imagens usadas nos experimentos foram fornecidas por LObe and Lung Analysis 2011 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) e VESsel SEgmentation in the Lung 2012 (RUDYANTO et al., Acessado em Maio, 2016).

Existem diversas técnicas de aquisição de imagens médicas, entre elas podemos mencionar imagens geradas através de aparelhos de raios-x, ultrassom, ressonância magnética, entre outros. Uma forma não invasiva e capaz de representar tridimensionalmente cada elemento que compõe o corpo humano com precisão, é a tomografia computadorizada (TC), a qual tem revolucionado o estudo em vivos. Na Figura 1.1, podemos ver uma amostra de uma imagem gerada por TC.

Imagens como, por exemplo, fotografias são chamadas de imagens bidimensionais por conter apenas duas dimensões, largura e altura. Imagens bidimensionais são compostas de pequenos pontos, chamados de pixel (do Inglês, *Picture Element*). Imagens de TC são

¹Pneumologia é a especialidade médica a qual é responsável pelo tratamento das patologias das vias aéreas inferiores. O profissional habilitado para atuar na mesma é denominado "pneumologista".

Figura 1.1: Imagens capturadas utilizando TC. (a) Visão tridimensional, (b) Fatia Axial, (c) Fatia Coronal e (d) Fatia Sagital.

compostas por várias imagens sobrepostas em camadas, onde cada camada representa os dados de uma fatia do corpo sendo examinado. Cada elemento da imagem, nesse caso, representa dados de um volume pertencente ao mundo real, o qual tem altura, largura e profundidade, caracterizando uma imagem tridimensional, representando um volume. Devido representar um volume, em imagens tridimensionais, os elementos são chamados de voxel (do Inglês, *Volume Element*).

Vizinhança de um elemento (pixel/voxel) pode ser definida como todos os elementos próximos a ele. No caso de imagens bidimensionais, como em fotografias, temos quatro vizinhos de aresta e quatro vizinhos de canto, porém em imagens tridimensionais devido o voxel ter formato cúbico ou paralelepipedal, temos oito vizinhos de canto, seis vizinhos de face e doze vizinhos de aresta, o que soma um total de vinte e seis elementos vizinhos

Figura 1.2: Vizinhança (a) de aresta, (b) de vértice e (c) face, (d) completa de um voxel.

a um voxel. Na Figura 1.2, podemos ver as vizinhanças principais de um pixel e de um voxel.

Existem muitas vantagens no uso de imagens geradas por TC para auxiliar diagnósticos, porém analisar imagens tomográficas pulmonares de forma manual é uma tarefa de alto consumo de tempo devido ao grande volume e complexidade para visualizarmos e analisarmos essa imagem pois voxels precisam ser vistos de forma tridimensional para que sejam analisados. Isto é, por ele representar um volume real, devemos verificar o volume próximo em todas as direções, o que nos dá até 26 voxels vizinhos a cada elemento, enquanto em imagens tradicionais, temos apenas 8 elementos vizinhos a cada elemento, como pode ser visto na Figura 1.2.

Figura 1.3: Casos problema encontrados no banco de dados de LOLA11. (a) Má formação do pulmão, (b) Pulmão com nódulos, (c) Artefatos e (d) Alto Ruído.

Anualmente, milhares de artigos são publicados que descrevem novos algoritmos a ser aplicados a imagens médicas e biomédicas, e vários novos produtos aparecem no mercado baseado em tais algoritmos. Porém, poucos artigos providenciam uma comparação justa e direta das novas soluções propostas com o estado de arte. *Grand Challenges in Biomedical Image Analysis* (GRAND..., Acessado em Setembro, 2016) encoraja organização de estudos comparativos e oferece um website onde pode ser disponibilizado código e conjunto de dados usados a fim de permitir que outros times possam comparar seus resultados de maneira justa usando as mesmas métricas.

Uma das competições organizadas como parte da *Grand Challenges in Biomedical*

Image Analysis (GRAND..., Acessado em Setembro, 2016), foi a competição LOLA 11 cujo objetivo é comparar métodos para segmentação automática e semiautomática de pulmões e lóbulos pulmonares de imagens de tomografia do torax.

Além das dificuldades que envolvem a segmentação pulmonar com imagens de TC, as imagens usadas para avaliação de acurácia podem ser consideradas desafiadoras devido ao grande número de imagens com quantidade expressiva de anomalias (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014).

Existem vários algoritmos automáticos para resolver este problema, mas com diferentes acurácias e consumos de tempo. Uma avaliação de acurácia foi feita como parte da competição LOLA11 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) em 55 imagens tomográficas.

1.1 Objetivo

Desejamos desenvolver um método simples e rápido para segmentação de tecidos pulmonares que possa ser utilizado por centros de radiologia a fim de facilitar o diagnóstico e obter maior precisão. Para alcançar nosso objetivo, devemos solucionar dois subproblemas.

O primeiro é a segmentação pulmonar, que consiste na identificação de regiões como fundo, traqueia, veias, pulmões esquerdo e direito, o qual é um processo dificultado pela presença de ruído, artefatos, baixo contraste, doenças e remoções cirúrgicas.

O segundo é a segmentação dos lóbulos pulmonares, onde o pulmão esquerdo deve ser dividido em dois lóbulos, superior e inferior, e o pulmão direito deve ser dividido em três lóbulos, superior, inferior e médio.

O critério de divisão pulmonar é a pleura, membrana que separa o pulmão em lóbulos cujo interior há uma fissura chamada cavidade pleural, a qual é preenchida por um fluido lubrificante chamado de líquido pleural.

Contudo a pleura é muito delicada e não é claramente visível nos exames de tomografia computadorizada, além da ocorrência de procedimentos de lobectomia (remoção cirúrgica de lóbulos), dificultam a segmentação dos lóbulos, então buscamos identificar as veias presentes no pulmão a fim de obter informação sobre lóbulos.

De acordo com o protocolo padrão de referência definido por LOLA11, o volume do pulmão inclui toda a região interna ao pulmão, o que inclui veias, brônquios e possíveis áreas degradadas por doenças. Já a traqueia deve ser considerada como fundo. Nosso

Figura 1.4: Divisão de lóbulos de acordo com as fissuras. Fonte: (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014)

objetivo é conseguir uma segmentação simples, rápida e precisa do pulmão e seus lóbulos.

1.2 Organização do trabalho

No Capítulo 2, detalharemos alguns dos trabalhos relacionados a área de segmentação de pulmões e também de lóbulos, explicando como outros métodos abordaram o assunto. No Capítulo 3, será detalhado o método proposto para segmentação de pulmões. No Capítulo 4, será apresentado nossa proposta para a segmentação de lóbulos. No Capítulo 5, serão apresentados e discutidos os resultados de nosso método.

2 Trabalhos Relacionados

Processamento digital de imagens é uma área que surgiu na década de 60 com a necessidade de se processar as imagens transmitidas por sondas da NASA. Nas últimas décadas, tem sido observado um grande crescimento nesta área. A área de processamento digitais de imagem trata do estudo de técnicas de manipulação, análise e armazenamento de imagens digitais.

Para efetuar processamento de imagens existe várias técnicas, algumas delas usadas em nosso algoritmo, por exemplo, segmentação e filtragem espacial, que serão explicadas no decorrer do Capítulo 3. Neste capítulo buscamos comentar métodos de outros grupos que participaram da competição LOLA11 a fim de compararmos nosso método com os demais.

A competição LOLA11 providencia imagens tomográficas do torax com anormalidades para as quais padrões de referência têm sido estabelecidos para segmentação de pulmões e lóbulos, assim obtendo uma comparação justa entre algoritmos usando método de *Maximum Overlap* contra o protocolo padrão de referência. Além disto, uma aproximação do tempo de execução também é fornecido no resumo dos trabalhos.

O método de *Maximum Overlap* consiste em testar o arquivo de segmentação contra o protocolo de referência e depois repetir o teste com o mesmo protocolo de referência transladado, isto é, movido alguns elementos em cada eixo. É usado como pontuação, o melhor valor, ou seja, o valor de maior sobreposição de elementos dentre os testes (BERG et al., 1996).

O padrão de referência foi produzido pela segmentação manual dos pulmões e lóbulos. Um conjunto de regras foi usada para indicar onde deixar a segmentação dos pulmões intersectar os vasos e brônquios no hilo pulmonar. Parte da árvore bronquial e de vasos sanguíneos distante destas interseções são considerados parte do volume pulmonar, traduzido de LOLA11 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014).

Nas próximas seções, um breve resumo dos algoritmos propostos por alguns dos grupos

que submeteram trabalhos utilizando LOLA11 na área de segmentação de pulmões e lóbulos.

2.1 Segmentação de pulmões

Na área de segmentação de pulmões, como parte do LOLA11, dezesseis outros grupos propuseram trabalhos, sendo quinze deles, algoritmos automáticos. Alguns dos algoritmos publicados por outros grupos serão descritos abaixo para que possamos comparar nosso método com trabalhos da literatura, o que será realizado na Seção 5, Discussão e Resultados.

- Human

O grupo Human não construiu um algoritmo computacional, ao invés disto, usa um segundo observador humano para, com auxílio de uma ferramenta, segmentar todas as imagens de forma não automática e estudar a variabilidade entre estas anotações e o padrão de referência do LOLA11.

Apesar do trabalho feito por este grupo não se tratar de um algoritmo automático para que possamos compará-lo com nosso método, esse grupo mostra qual a possível variação na percepção humana seguindo precisamente as instruções do protocolo de referência, por observadores diferentes.

Por este grupo não desenvolver um método automático ou semiautomático, objetivo da competição LOLA11 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014), não foi publicado nenhum artigo contendo uma descrição detalhada sobre o procedimento deste grupo.

- Método de Soliman et al.

Soliman et al. (SOLIMAN et al., 2015), utiliza uma base de dados com dados de vários pacientes para os quais já existe um resultado de segmentação, assim o procedimento inicia registrando a nova imagem a ser segmentada, sujeito de teste, no banco de dados de treinamento a fim de obter um mapeamento inicial que fornece os valores de deformação para cada voxel.

Imagens diferentes podem ter muitas características semelhantes quando vistas sob um nível de detalhamento menor. Ao aplicar uma filtragem espacial gaussiana (GONZALEZ; WOODS, 2002), obtemos uma imagem leve ou fortemente borrada, isto é, onde alguns detalhes são suprimidos da imagem.

Figura 2.1: Espaço de escala gaussiana $L(x, y, t)$ com tamanho de kernel variando entre zero e vinte e cinco. (a) Tamanho zero, correspondendo a imagem original, (b) cinco, (c) quinze e (d) vinte e cinco.

Ao realizar a filtragem espacial gaussiana com núcleos (kernel) de diferentes tamanhos, sobre a mesma imagem, obtemos o espaço de escala gaussiana (GONZALEZ; WOODS, 2002), que consiste em um conjunto da imagem original filtrada com kernels de diferentes tamanhos. Na Figura 2.1, podemos ver um exemplo de espaço de escala gaussiana criado com 4 diferentes níveis de detalhamento obtidos usando filtragem gaussiana.

Para a imagem original é gerado o espaço de escala gaussiana a fim de com esses dados realizar uma correlação entre o sujeito de teste e cada sujeito no banco de dados de treinamento, um banco de dados com imagens já segmentadas, o que permite selecionar os sujeitos mais similares no banco de dados de treinamento.

Com os sujeitos similares identificados buscando imagens mais semelhantes através do espaço de escala gaussiana é calculado a probabilidade a posteriori de ocorrência

de cada voxel sobre o modelo, o que gera uma estimativa da aparência.

O campo aleatório de Markov - MRF (do Inglês, *Markov Random Field*) é um modelo que descreve um conjunto de variáveis aleatórias interligadas por relações de dependência. A melhor forma de visualizar essa estrutura de dependências é um grafo não direcionado que quando a probabilidade de junção entre duas variáveis é estritamente positiva, caracteriza um campo aleatório de Gibbs, sendo chamado de campo aleatório de Markov-Gibbs - MGRF (do Inglês, *Markov-Gibbs Random Field*) como pode ser visto na Figura 2.2.

Figura 2.2: Exemplo de MRF modelado usando grafo. As arestas indicam dependência entre as variáveis que são modeladas como nós.

Os modelos MRF e MGRF têm sido utilizados no processamento de imagens sendo geralmente formulado a partir de uma imagem original segmentada, construindo-se um grafo de regiões adjacentes - RAG (do Inglês, *Region Adjacent Graph*), onde dois nós têm conectividade entre eles se duas regiões compartilham uma fronteira. Assim, é estimado uma probabilidade *a posteriori* - MAP (do Inglês, *Maximum a posteriori*) para estimar a probabilidade de união entre duas regiões vizinhas.

O MAP, ao contrário de outras distribuições de probabilidade, considera a probabilidade de um evento ocorrer novamente dado os eventos que já ocorreram. Neste caso, baseado em algumas imagens segmentadas, é calculada a probabilidade de cada região ser segmentada novamente com a mesma união entre regiões. Então, o MAP identifica a combinação linear de gaussianas dominante para cada classe e usa o mapeamento de regiões inicial para identificar o MGRF e, por fim, realiza a segmentação das regiões pulmonares usando as informações obtidas pelos sujeitos similares combinadas no MGRF.

O método em questão obteve até o presente momento o melhor valor de segmentação automática em LOLA11 usando o método de *Maximum Overlap* (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014). Podemos ver no Algoritmo 1, podemos ver como é construído o modelo adaptativo baseado em aparência e no

Algoritmo 2, temos a descrição de como os pulmões são segmentados.

Algoritmo 1 Algoritmo para criação do modelo adaptativo baseado em aparência (SOLIMAN et al., 2015)

1. Registre o sujeito de teste no banco de dados de treinamento para conseguir os campos de deformação para cada voxel.
 2. Gere os dados de espaço de escala gaussiana (do inglês GSS).
 3. Faça uma correlação de similaridade cruzada entre o sujeito de teste e cada sujeito no banco de dados de treinamento e seleciona os mais similares.
 4. Estime o modelo de forma baseado em aparência calculando o valor da probabilidade da forma prévia sobre cada voxel usando as seguintes etapas para os dados obtidos pelo espaço de escala gaussiana e dados originais.
-

Algoritmo 2 Algoritmo para segmentação de pulmões (SOLIMAN et al., 2015)

1. Ache um mapeamento inicial pela classificação bayesiana de voxels da imagem tomográfica usando probabilidades estimadas pelo algoritmo de estimação de forma baseado na aparência.
 2. Estime o modelo de intensidade condicional identificando os modelos LCDG de sinais de cada objeto representado por um dos modos dominantes.
 3. Use o mapa de região inicial para identificar o modelo MGRF do mapeamento das regiões.
 4. Faça a segmentação bayesiana final das regiões do pulmão usando a junção do modelo MGRF.
-

- Método de Lassen et al.

Lassen et al. (LASSEN et al., 2011), inicia uma análise em baixa resolução para encontrar os pulmões, isto é, considerando apenas parte dos elementos da imagem original é simulado uma imagem de menor resolução.

E, a partir destes dados com a localização dos pulmões, encontra um *threshold* apropriado e executa um algoritmo de crescimento de regiões na imagem em alta resolução, resultando em uma segmentação do sistema respiratório.

Estruturas semelhantes a traqueia são encontrados usando uma forma 2D com análise de tamanho e o candidato mais provável é usado. Depois, alguns métodos são usados para remover vazamentos, reduzir ruído e melhorar a separação usando watershed e um algoritmo de fechamento para incluir vasos sanguíneos e patologias.

- Método de Weinheimer et al.

Weinheimer et al. (WEINHEIMER et al., 2011), inicia buscando pela árvore traqueia bronquial. Depois um algoritmo de crescimento de regiões 3D buscando por estruturas de baixa densidade e um algoritmo de preenchimento de buracos para incluir estruturas de alta densidade.

A separação de pulmões é feita se apenas um componente conectado é encontrado, nestes casos, usando um método para buscar um caminho mínimo separando os pulmões.

- Método de Pinho et al.

Pinho et al. (PINHO; DELMON; VANDEMEULEBROUCKE, 2011), executa 6 tipos de operações: extração do paciente, localização da traqueia, crescimento de regiões adaptativo, segmentação baseada em corte de grafo, crescimento de regiões e operações morfológicas.

A extração do paciente é uma operação responsável por remover da imagem estruturas externas ao corpo do paciente. Após a localização da traqueia na imagem restante, o crescimento de regiões adaptativo identifica os pulmões ao crescer uma região a partir dos pontos terminais da traqueia.

Após a identificação dos pulmões, eles podem tocar-se, sendo necessário o algoritmo de corte em grafo, novo crescimento de regiões e execução de operações morfológicas para finalizar a segmentação.

- Método de Korn et al.

Korn et al. (KORN et al., 2011), usou *Cognition Network Technology* implementado na plataforma Definiens XD (DEFINIENS..., Acessado em Setembro, 2016) e executa basicamente 4 etapas para segmentar pulmões.

1. A segmentação inicial retorna uma segmentação aproximada para pulmões patológicos e não patológicos.
2. Refinamento da borda do pulmão refina a borda para órgãos vizinhos.
3. Segmentação da traqueia e brônquio no contexto destes objetos.
4. Refinamento da separação entre pulmão esquerdo e direito, o que é necessário para corrigir a segmentação quando os pulmões estão unidos.

- Método de Sun et al.

Sun et al. (SUN; BAUER; BEICHEL, 2011) usa um algoritmo de detecção de costelas para iniciar um modelo e grosseiramente segmentar os pulmões. Depois, o modelo

é adaptado para os dados da imagem por meio de um algoritmo que encontra uma superfície ótima.

- Método de Gu et al.

Gu et al. (GU; PU, 2011) executa uma segmentação de pulmões convencional e busca por erros. Se erros forem encontrados, inicia uma busca por marcas chaves e pontos importantes como costelas, linhas abaixo dos pulmões e bordas entre pulmões.

Em seguida um método de regulação elástica é aplicado para ajustar um template predefinido.

- Método de Hosseini et al.

Hosseini et al. (HOSSEINI-ASL; ZURADA; EL-BAZ, 2011) primeiro preprocessa a imagem, o exterior do corpo humano é removido, depois modelando a imagem, uma matriz 4D é gerada incluindo para cada voxel, os sinais de sua vizinhança. A segmentação é baseada no algoritmo KMeans.

2.2 Segmentação de lóbulos pulmonares

Dois grupos apresentaram solução para a segmentação automática de lóbulos pulmonares na competição LOLA11. Um deles não dispomos de dados sobre seu método pois até a presente data, o artigo com sua descrição está indisponível no website da competição LOLA11. O outro grupo, Lassen et al. (LASSEN et al., 2011) combina informação de fissuras, vasos sanguíneos e árvore brônquica, conjunto formado pelos dois brônquios principais situados fora dos pulmões e suas ramificações nos dois pulmões, em uma única imagem a ser usada pelo algoritmo de watershed.

Pressupondo que não existem vasos sanguíneos mais espessos próximo às bordas de lóbulos (Figura 2.3b), é calculada a distância de cada voxel para o vaso sanguíneo mais próximo, o que gera valores mais altos de distância próximo às bordas de lóbulos onde encontra-se a pleura visceral, membrana que separa os pulmões e tem alta densidade comparado a outros tecidos que recobrem o pulmão, consequentemente tendo maior intensidade de cor, o que pode ser visto na Figura 2.3e.

Os lóbulos são separados por uma fina membrana que pode ter apenas parte de seu volume visível, aumentando a dificuldade em diferenciá-la de possível ruído. Para reduzir o ruído, primeiro é aplicada uma suavização gaussiana, porém este procedimento torna parte da fissura borrada e com menor visibilidade. Então um algoritmo de melhoramento

Figura 2.3: Fluxograma descrevendo algoritmo. Fonte: Lassen et al. (LASSEN et al., 2011).

é usado para analisar os vizinhos do voxel a fim de melhorar a visibilidade de elementos presentes na pleura.

Se o voxel está cercado por voxels claros em pelo menos três dos planos de corte entre os nove planos que cortam um voxel, uma quantidade relativa ao brilho da estrutura ao redor é proveniente da intensidade dos voxels centrais, sendo então usada essa informação para melhorar a detecção da fissura. Veja mais detalhes na Figura 2.4. O resultado desse algoritmo pode ser visto em (Figura 2.3d).

Após melhorar o contraste da fissura, um algoritmo baseado em análise de autovalores obtidos a partir da matriz hessiana detecta parte da Pleura, o que pode ser visto na Figura 2.3g e as informações obtidas são combinadas em uma imagem de custo a ser usada no procedimento de *watershed*, que modela o despejar de água sobre a imagem tomando a intensidade de cor como uma dimensão extra simbolizando a profundidade do ponto como em um mapa topográfico, como detalhado na Seção 4.3, onde utilizamos este algoritmo.

Figura 2.4: Brilho de elementos no entorno e os planos cortando um voxel utilizados na detecção da fissura. Fonte: Lassen et al. (LASSEN et al., 2011).

3 SALSA – Simple Automatic Lung Segmentation Algorithm

O SALSA foi construído com o propósito de buscar novos métodos de segmentação para lidar com a segmentação de imagens tomográficas pulmonares. Para criação do algoritmo foram buscados algoritmos simples que, quando combinados, produzam bom resultado quando comparados a outros trabalhos relacionados a mesma área de segmentação.

Buscamos implementar uma alternativa independente de softwares comerciais com o intuito de criar um algoritmo gratuito, rápido e com baixos requisitos de hardware. Nosso software busca facilitar o processamento das imagens e ao mesmo tempo produzir uma solução simples que possa ser implementada e aprimorada com facilidade pela comunidade.

O método proposto foi publicado no *XX Iberoamerican Congress on Pattern Recognition* - CIARP 2015 (COSTA; CARVALHO, 2015) e estamos melhorando os resultados alcançados pelo algoritmo para realizar nova publicação.

O algoritmo usado como núcleo do projeto foi o algoritmo FloodFill (GONZALEZ; WOODS, 2002) em sua versão 3D para colorir a imagem baseado tanto em intensidade, cor de pintura e vizinhança descrito no Algoritmo 3.

Algoritmo 3 FloodFill2D com vizinhança de aresta

```

1: função FLOODFILL2D(x:inteiro , y:inteiro , corVelha:inteiro , corNova:inteiro)
2:    $cor \leftarrow getColor(x, y)$ 
3:   se  $cor = corVelha$  então
4:      $setColor(x, y, corNova)$ 
5:     FloodFill2D( $x - 1, y, corVelha, corNova$ )
6:     FloodFill2D( $x + 1, y, corVelha, corNova$ )
7:     FloodFill2D( $x, y - 1, corVelha, corNova$ )
8:     FloodFill2D( $x, y + 1, corVelha, corNova$ )

```

O algoritmo FloodFill é um algoritmo bastante simples e eficiente para pintura de grandes regiões. Sua implementação padrão baseia-se apenas na informação de cor, porém

pode facilmente ser adaptado para incluir outras condições.

O funcionamento do algoritmo baseia-se em, a partir de um ponto inicial, substituir sua cor original por uma nova e, recursivamente, todos os vizinhos a este elemento inicial com a cor original são pintados com a nova cor até que não existam elementos com a cor original vizinhos à região pintada. Um exemplo bastante comum do uso deste algoritmo é na ferramenta balde de tinta comumente encontrada em softwares editores de imagem.

Figura 3.1: Fluxo geral do SALSA. (a) Imagem Original, (b) Pré-processamento da imagem, (c) Filtro de Sobel, (d) Detecção da borda, (e) Remoção do background, (f) Segmentação da traqueia, (g) Corte de Pulmões, (h) Segmentação de vasos, (i) Segmentação final.

No Algoritmo 3 encontra-se uma versão 2D do FloodFill que usa vizinhança de aresta. Neste trabalho, foi usada a vizinhança de face e foram adicionados cláusulas extras além da cor e vizinhança para definir os critérios de pintura como, por exemplo, para borda pulmonar, os elementos devem estar na fronteira entre branco e cinza, onde é analisado a intensidade do elemento ao invés de cor de classe a que pertence.

Os tomógrafos computadorizados são ajustados a fim de que a intensidade obtida no processo seja calibrada de forma a que ossos densos tenham intensidade de 1000 HU (do Inglês, *Hounsfield Units*), a água tenha intensidade 0 HU e o ar deve ter intensidade de -1000 HU, conforme Tabela 1.

Tecido	HU
Osso denso	1000
Músculo	50
Matéria Branca	45
Matéria Cinzenta	40
Sangue	20
Fluido Cerebroespinal	15
Água	0
Gordura	-100
Pulmão	-200
Ar	-1000

Tabela 1: Escala Hounsfield

Durante o processamento, algumas etapas são necessárias. Na Figura 3.1 podemos ver uma descrição resumida do fluxo de execução do algoritmo usado para segmentação de pulmões.

a - Pré-processamento da imagem

Durante este processo é executado um algoritmo de *thresholding* a fim de descartar dados desnecessários e, além disto, é calculado uma estimativa do nível de ruído presente na imagem e, através de um algoritmo baseado em autômatos finitos, são identificadas algumas sementes.

As sementes são voxels que servem de ponto de partida para execução de outros algoritmos. Elas estão presentes na fronteira do pulmão e são necessárias para a execução da etapa seguinte. Estas sementes são encontradas com uso de um algoritmo baseado em autômatos finitos.

b - Filtro de Sobel

Caso a imagem possua pouco ruído, é aplicado um filtro de Sobel a fim de detectar

contornos de borda adicionais na parte interna do pulmão. Caso contrário, nenhuma filtragem é aplicada para melhorar a borda. Sendo usado no último caso apenas a detecção de borda padrão.

c - Detecção da borda

Um algoritmo de busca por borda consiste em, a partir de uma semente inicial, expandir uma região baseado em um critério que define a borda a partir deste elemento inicial. Assim, buscamos elementos de cor cinza que tenham vizinhos brancos e acrescentamos a esta fronteira, elementos de borda encontrados através do filtro de Sobel calculado na etapa anterior.

d - Remoção do background

Para a remoção de background buscamos regiões externas ao pulmão em cada fatia axial evitando processamento tridimensional. Isto é feito para que nós não adicionemos ao background estruturas como veias que por ter densidade próxima a estruturas presentes no background e ser externos a borda, caso fosse efetuado processamento tridimensional, seriam segmentadas como background. Depois buscamos elementos do background na região hilar, que podem ter ficado oclusos no processamento 2D efetuado em fatias axiais.

e - Segmentação da traqueia

A traqueia é encontrada buscando nos elementos ainda não segmentados, ou seja, elementos que ainda não foram atribuídos a uma classe, por fatias axiais com baixa área conexas, estabelecendo taxa de variação máxima em sua área a cada nova fatia.

f - Corte de Pulmões

Após os passos anteriores, toda a região não segmentada é de pulmão ou vasos sanguíneos. Dividimos, então, o volume ao meio com um corte planar e preenchemos os volumes de cada lado usando sementes identificadas ao fim da traqueia. Em seguida analisamos o corte para verificar onde apenas um dos lados do corte está colorido a fim de segmentar áreas ainda não segmentadas.

g - Segmentação de vasos

Agora que já temos uma segmentação inicial do pulmão, todas as áreas não segmentadas são vasos sanguíneos ou possíveis áreas mal segmentadas devido às bordas muito fechadas usando o Sobel. Para detectar vasos sanguíneos, nessa etapa apenas buscamos áreas pequenas próximo as bordas internas do pulmão para demarcar como vasos, os quais serão anexados como parte do pulmão na etapa final.

h - Segmentação final

Os voxels restantes fora da borda, não afetados pela remoção de background, são segmentados como vasos sanguíneos e unidos ao pulmão para produzir o resultado final.

3.1 Pré-processamento da imagem

Ainda enquanto a imagem é carregada para a memória, algumas operações importantes são executadas. Entre elas, o algoritmo de *thresholding*, detecção de sementes e detecção de ruído, necessário para que seja tomado importantes decisões nos demais algoritmos. De forma geral, a leitura e pré-processamento da imagem se comporta como demonstrado no Algoritmo 4, o qual tem funções Thresholding, changeState, updateDepth e isNoisy, detalhadas ao longo desta seção.

Algoritmo 4 Ler arquivo de imagem

```

1: função LOADFILE(image : voxel***)
2:   state  $\leftarrow$  Outside
3:   potentialSeeds  $\leftarrow$   $\emptyset$ 
4:   para todo voxel  $\in$  image faça
5:     originalIntensity  $\leftarrow$  getIntensity(voxel)
6:     thresholdedIntensity  $\leftarrow$  Thresholding(originalIntensity)
7:     SetIntensity(voxel, thresholdedIntensity)
8:     se IsNoisy(voxel) então
9:       noisyElements  $\leftarrow$  noisyElements + 1
10:    intensity  $\leftarrow$  getIntensity(voxel)
11:    UpdateDepth (state, intensity, depth)
12:    ChangeState(voxel, state, depth, potentialSeeds)
13:  se noisyElements > MaxNoise então
14:    para todo voxel  $\in$  image faça
15:      thresholdedIntensity  $\leftarrow$  getIntensity(voxel)
16:      se thresholdedIntensity > -474HU então
17:        SetIntensity(voxel, -474HU)
18:  Estimar área da fatia axial usando sementes em potencial
19:  Escolha fatia com maior área estimada
20:  Retorne sementes mais distantes sobre eixo X na fatia escolhida

```

3.1.1 Thresholding

Thresholding é um processo de binarização de uma figura, onde há o descarte de cores baseado em determinado limiar, sendo cores mais claras que o limite escolhido, marcadas como branco e cores abaixo desse limite, como pretas, por exemplo.

Descartar cores quando se trata de processamento de imagem significa perder características da imagem, o que em geral não traz benefícios, porém quando trata-se de imagem de tomografia computadorizada, os tomógrafos são calibrados para que as imagens geradas tenham, para o ar, intensidade de -1024 HU e para a água, intensidade de 0 HU, enquanto estruturas ósseas rígidas, +1024 HU. Assim, estamos descartando apenas dados desnecessários por se tratar de dados sobre tecidos irrelevantes para a segmentação pulmonar. Veja como é feito o procedimento no Algoritmo 5.

Algoritmo 5 Thresholding da imagem

```

1: função THRESHOLDIND(originalIntensity:inteiro)
2:    $MinHU \leftarrow -1024$ 
3:    $MaxHU \leftarrow -239$ 
4:   se  $originalIntensity > MaxHU$  então
5:     retorna  $MaxHU - MinHU$ 
6:   se  $originalIntensity < MinHU$  então
7:     retorna 0
8:   retorna  $originalIntensity + MaxHU - MinHU$ 

```

Devido à presença de artefatos e aos tomógrafos descalibrados, muitas vezes podemos ter valores fora dessa escala. O processo de threshold não é capaz de remover essas influências negativas, mas é capaz de diminuí-las.

No caso de processamento de tecidos pulmonares, estamos interessados em estudar vias aéreas e tecidos esponjosos, então enquanto a imagem original (Figura 3.2a) é carregada para a memória, é aplicado um algoritmo de threshold (Figura 3.2b) a fim de descartar dados desnecessários, melhorando o contraste. Para isso, são mantidas como intensidade mínima e máxima, respectivamente, as intensidades de -1024 HU, que trata-se da intensidade do ar, até -239 HU, que está relacionado a intensidade de tecidos esponjosos. Valores fora dessa faixa são guardados com o valor mínimo ou máximo.

Figura 3.2: Imagem Axial antes e depois da aplicação do Threshold.

3.1.2 Detecção de sementes

Ainda enquanto a imagem é lida, um algoritmo baseado em autômato finito, um algoritmo baseado em máquina de estados que valida uma sequência de símbolos, busca por sementes usando os dados calculados pelo Threshold.

Figura 3.3: Sementes detectadas. (a) Imagem obtida após o processo de *thresholding* e (b) Sementes identificadas.

O algoritmo recebe uma cadeia de símbolos, os dados de uma linha sobre o plano axial, tentando achar dois voxels com a maior quantidade de pulmão possível entre eles o que

pode ser estimado baseado na intensidade da imagem e estados que limitam a estimativa às regiões internas ao corpo humano. Os estados do algoritmo controlam a largura e altura do pulmão em cada fatia. Veja abaixo, na Figura 3.3, a entrada e saída do algoritmo.

Para criar o algoritmo responsável pelo processamento exibido na Figura 3.3, foi criado um algoritmo baseado no algoritmo Scanline (GONZALEZ; WOODS, 2002), usado para preenchimento de polígonos em computação gráfica. A ideia do algoritmo Scanline é, usando-se estados, identificar se está sendo pintada uma área interna ou externa do polígono mudando de estado entre interno e externo ao polígono ao cruzar uma borda.

Figura 3.4: Automato Finito usado para detectar sementes na borda pulmonar, Fluxograma geral do algoritmo.

Como pode ser visto, considerando que a borda seja qualquer elemento na fronteira entre branco e outra cor, obtemos uma imagem com bordas bem definidas, o que a partir de algum conhecimento anatômico, podemos facilmente definir significado para as bordas em sua respectiva ordem, por exemplo, ao encontrar uma borda seguida de muitos elementos brancos, provavelmente estará na região mais externa do corpo humano, próximo a pele. Assim, propomos o autômato exemplificado na Figura 3.4.

Algoritmo 6 Busca por sementes

- 1: **função** CHANGESTATE(voxel:voxel, state:string , depth:inteiro , potentialSeeds:voxel*)
- 2: $x \leftarrow \text{getX}(\text{voxel}); y \leftarrow \text{getY}(\text{voxel}); z \leftarrow \text{getZ}(\text{voxel})$
- 3: **se** $\text{depth} = 10$ **então**
- 4: **se** $\text{state} = \text{Outside}$ **então**
- 5: $\text{depth} \leftarrow 0; \text{state} \leftarrow \text{Body}$
- 6: **senão**

```

7:         se  $state = Body$  então
8:              $potentialSeeds \leftarrow \{potentialSeeds\} + seed(x - depth, y, z)$ 
9:              $depth \leftarrow 0; state \leftarrow Lung$ 
10:        senão
11:            se  $state = Lung$  então
12:                 $potentialSeeds \leftarrow \{potentialSeeds\} + seed(x - depth, y, z)$ 
13: função UPDATEDDEPTH( $state:string$  ,  $intensity:inteiro$  ,  $depth:inteiro$ )
14:     se  $state = Outside \mid state = Lung$  então
15:         se  $intensity < MaxValue$  então
16:              $depth \leftarrow 0$ 
17:         senão
18:              $depth \leftarrow depth + 1$ 
19:     senão
20:         se  $intensity < MaxValue$  então
21:              $depth \leftarrow depth + 1$ 
22:         senão
23:              $depth \leftarrow 0$ 

```

No Algoritmo 6 são descritos as Funções *ChangeState* e *UpdateDepth*, usadas no Algoritmo 4. Estas funções são chamadas em sequência sempre que é lido um novo voxel presente na imagem a fim de identificar se estes elementos estão em fronteira com o pulmão conforme exemplificado na Figura 3.4.

Na Linha 1, temos a definição da função *ChangeState* que recebe como parâmetros, respectivamente, um voxel que foi lido da imagem, o estado atual do autômato, o número de elementos seguidos dentro de um estado e um apontador para um vetor com as sementes encontradas. Na linha seguinte obtemos a posição X, Y e Z do voxel analisado.

Na Linha 3, vemos que qualquer mudança de estado só é efetuada caso encontre dez elementos seguidos no estado, isto ocorre porque na imagem tomográfica, cada voxel tem aproximadamente 0.71 milímetros, tendo então 10 voxels aproximadamente 7-8 milímetros o que é uma espessura menor que o de uma costela.

Nas Linhas 4 e 5, efetuamos a mudança de estado de externo ao corpo para interno ao corpo caso tenha sido encontrado dez elementos seguidos na cor branco, isto é, o valor máximo após o *thresholding*. Isto ocorre pois o interior ao corpo humano é formado basicamente por água e ossos, tendo estes elementos recebido o valor de intensidade mais alto após o *thresholding*.

Nas Linhas 6 a 9, modificamos o estado de interior ao corpo humano para interior aos pulmões, nesse caso isto ocorre ao encontrar dez elementos na cor cinza, isto é, elementos que não têm o valor máximo de intensidade após o *thresholding*. Devido ao pulmão ser preenchido com ar e tecidos pouco densos, a intensidade destes elementos nunca devem alcançar o valor máximo pois isto significaria água nos pulmões entre outras doenças.

Por fim, nas Linhas 10 a 12, se já está no estado pulmão e encontrou novos dez elementos brancos, significa que encontrou novo local de transição de elementos cinzentos, tecidos pulmonares, para interior ao corpo humano, tecido denso de músculos e ossos. Neste caso é atualizado a segunda semente encontrada na linha para este local de transição que marca o final do pulmão.

Na Função *UpdateDepth*, temos na Linha 13 a definição da função que recebe, respectivamente, o estado atual, a intensidade do último elemento lido na imagem e a quantidade de elementos no estado atual.

Nas Linhas 14 a 18 ocorre a atualização da quantidade de elementos para os estados externo ao corpo e interno aos pulmões que, para ambos os casos, o quantitativo de elementos deve simbolizar o número de elementos brancos nestes estados, assim, nas Linhas 15 e 16 esse quantitativo é zerado caso encontre um elemento cinza e incrementado nas Linhas 17 e 18 ao encontrar um elemento branco.

Nas Linhas 19 a 23 ocorre a atualização no quantitativo de elementos para o estado interno ao corpo, onde procuramos elementos cinzentos, tecido pulmonar, sendo incrementado o quantitativo ao encontrar estes elementos nas Linhas 20 e 21 e zerado em caso contrário.

O algoritmo exemplificado na Figura 3.4 e Algoritmo 6 processa as linhas presentes na fatia axial e identifica zero ou duas sementes por linha. Durante a execução, se ambas sementes encontradas em linhas vizinhas forem conexas entre si, essa informação é armazenada como uma região, onde pode-se estimar a área. Por fim, as sementes da linha de maior comprimento na região com maior área são selecionadas como sementes para a busca por borda.

3.1.3 Filtragem espacial

Em processamento de imagens, uma imagem é uma função com duas variáveis $f(x, y)$ que define a cor do pixel (x, y) e, para melhorar a qualidade ou extrair informações, podemos combinar essa imagem com um filtro gerando uma nova imagem. Para isso,

calculamos a multiplicação ponto a ponto entre a imagem e um filtro G onde o número de linhas e colunas do filtro definem o tamanho da janela do filtro, por exemplo na Equação 3.1. Esse procedimento é chamado de convolução, um operador linear que, a partir de duas funções dadas, resulta em uma terceira função que mede a superposição das funções dadas.

$$\begin{aligned} (f \otimes g)(x, y) &= \begin{bmatrix} f(x-1, y-1) & f(x, y-1) & f(x+1, y-1) \\ f(x-1, y) & f(x, y) & f(x+1, y) \\ f(x-1, y+1) & f(x, y+1) & f(x+1, y+1) \end{bmatrix} \otimes \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \\ &= 4 * f(x, y) - f(x, y-1) - f(x-1, y) - f(x+1, y) - f(x, y+1) \end{aligned} \quad (3.1)$$

De maneira geral, assumindo que o filtro G_{mn} tenha número ímpar de elementos, podemos definir a operação de convolução discreta $\otimes$ como na Equação 3.2.

$$(f \otimes G)(x, y) = h(x, y) = \sum_{k=\lfloor -m/2 \rfloor}^{\lfloor m/2 \rfloor} \sum_{l=\lfloor -n/2 \rfloor}^{\lfloor n/2 \rfloor} f(x+k, y+l) * g(k, l), \text{ se } (x+k, y+l) \in f \quad (3.2)$$

Filtragem espacial é um processo bastante conhecido em processamento de imagens e consiste de, a partir de uma imagem, aplicar um filtro através de convolução para ressaltar alguma característica de interesse da imagem. Por exemplo, para detectar contornos em imagens, um dos filtros mais usados é o filtro de Sobel, porém é um filtro bastante sensível a ruído, como pode ser visto na Figura 3.5.

O filtro de Sobel é usado para extrair informação das bordas presentes na imagem. Ele calcula uma aproximação do gradiente da intensidade da imagem em cada ponto, dando a direção da maior variação de luminosidade em cada ponto e uma quantização dessa variação. O filtro de Sobel pode ser definido como na Equação 3.3, sendo g_x uma aproximação da derivada parcial em x e g_y uma aproximação em y , podemos definir a magnitude do gradiente como $G = \sqrt{g_x^2 + g_y^2}$ e aplicá-lo a imagem usando a Equação 3.2, ou de forma aproximada calcular o filtro de Sobel como $f \otimes G = g_x \otimes A + g_y \otimes A$.

$$g_x(x, y) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad g_y(x, y) = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (3.3)$$

Figura 3.5: Aplicação de filtros sobre imagem. (a) Imagem axial com pouco ruído Lola11-05, (b) Imagem axial com muito ruído Lola11-07, (c) Sobel sobre Lola11-05 e (d) Sobel sobre Lola11-07.

Para que seja feito o processo de filtragem espacial usando o filtro de Sobel, o qual é muito sensível a ruído, precisamos estimar o nível de ruído na imagem a fim de aplicar o filtro de sobel somente caso a imagem seja compatível. Assim, após o *thresholding* realizado na etapa anterior, um algoritmo de estimação de ruído é aplicado e dependendo do resultado é então aplicado o filtro de Sobel.

Para estimar o ruído, identificamos a região do corpo humano externa aos pulmões onde, pelo nosso algoritmo de *thresholding*, todos os voxels deveriam ter cor branca e contamos quantos elementos não brancos existem. Precisamos obter com esse método

apenas uma estimativa de ruído por volume de tecido, não sendo necessário detecção precisa pois a variação entre as imagens afetadas se dá em ordem de grandeza. O algoritmo usado para estimação de ruído pode ser visto no Algoritmo 7.

Caso não seja detectado ruído excessivo, novo thresholding é aplicado limitando a escala para -1024 HU a -474 HU. O objetivo da redução na escala de intensidades é melhorar a diferenciação entre pulmão e órgãos próximos evitando, por exemplo, que o esôfago seja considerado como parte do pulmão no momento que ambos os órgãos se tocam.

Algoritmo 7 Estimação de nível de ruído

```

função ISNOISY(voxel : voxel)
  whiteVoxels  $\leftarrow$  0
  grayVoxels  $\leftarrow$  0
  para todo neighbor  $\in$  neighbors(voxel) faça
    se getIntensity(neighbor) = MaxValue então
      whiteVoxels  $\leftarrow$  whiteVoxels + 1
    senão
      grayVoxels  $\leftarrow$  grayVoxels + 1
  retorna whiteVoxels > grayVoxels & getIntensity(voxel) < MaxValue

```

A partir da execução do Algoritmo 7, temos uma quantização do ruído. Definimos então um limiar para definir uma quantidade de ruído aceitável para aplicação do filtro de Sobel 2D. Foi tentado utilizar o filtro de Sobel expandido para três dimensões, porém devido o voxel não ser isotrópico, o resultado foi melhor obtido com o Sobel 2D. A definição deste limiar não precisa ser precisa pois a variação para imagens com nível de ruído não aceitável ocorre em duas ordens de grandeza do limiar usado.

Algoritmo 8 Filtro de Sobel

```

função SOBELFILTER(image : voxel * **)
  para todo voxel  $\in$  image faça
    x  $\leftarrow$  getX(voxel)
    y  $\leftarrow$  getY(voxel)
    z  $\leftarrow$  getZ(voxel)
    Gx  $\leftarrow$  0
    Gy  $\leftarrow$  0

```

```

para  $i = -1, i \leq 1, i++$  faça
     $D_x = \text{getIntensity}(\text{image}[x+1][y+i][z]) - \text{getIntensity}(\text{image}[x-1][y+i][z])$ 
     $D_y = \text{getIntensity}(\text{image}[x+i][y-1][z]) - \text{getIntensity}(\text{image}[x+i][y+1][z])$ 
    se  $i = 0$  então
         $G_{x+} = 2 * D_x$ 
         $G_{y+} = 2 * D_y$ 
    senão
         $G_{x+} = D_x$ 
         $G_{y+} = D_y$ 
     $\text{setSobel}(\text{voxel}, G_x + G_y)$ 

```

Além da redução no valor de threshold, caso exista pouco ruído, é realizado um filtro de Sobel em todos os voxels da imagem a fim de detectar bordas, como exemplificado no Algoritmo 8. Caso exista muito ruído, o filtro de Sobel não é aplicado pois é muito sensível a ruído. Na Figura 3.5c, podemos ver o resultado da aplicação do filtro de Sobel em uma figura com pouco ruído e na Figura 3.5d, a aplicação do mesmo filtro em imagem com muito ruído. Caso exista ruído excessivo também é ajustado a definição de vizinhança usada pelo algoritmo para detecção de borda.

3.2 Busca por borda

As sementes encontradas pelo autômato finito são usadas para fazer uma busca por borda (GONZALEZ; WOODS, 2002) isolando o sistema respiratório (Figura 3.6a) baseado na informação de intensidade dos elementos como pode ser visto no Algoritmo 9.

A busca por borda é expandida com a busca por voxels conexos à borda inicial, utilizando os valores mais claros encontrados pelo filtro de Sobel (Figura 3.6b - vermelho) e vizinhança com proporção de voxels cinza adequada de acordo com o nível de ruído (Figura 3.6b - preto), necessário para garantir o fechamento da borda.

Caso a imagem seja ruidosa, apenas a informação de vizinhança é usada, sendo desprezado os valores do filtro de Sobel e efetuando uma borda mais restritiva para evitar incluir voxels ruidosos externos ao pulmão.

Algoritmo 9 Algoritmo de Busca por Borda

- 1: **função** INTENSITYBASEDBOUNDARYSEARCH(BorderSeeds:voxel*)
- 2: **para todo** $\text{voxel} \leftarrow \text{popBack}(\text{BorderSeeds})$ **faça**

```

3:   se getColor(voxel)  $\neq$  CorBorda então
4:     flag  $\leftarrow$  false
5:     whiteElements  $\leftarrow$  0
6:     grayElements  $\leftarrow$  0
7:     para todo element  $\in$  neighbors(voxel) faça
8:       se thresholdedIntensity(element) < MaxThreshold então
9:         grayElements  $\leftarrow$  grayElements + 1
10:    senão
11:      whiteElements  $\leftarrow$  whiteElements + 1
12:  se whiteElements > 0 & grayElements > MinElements então
13:    paint(voxel, CorBorda)
14:    pushBack(BorderSeeds, neighbors(voxel))

```

Figura 3.6: Detecção de borda. (a) Visão Tridimensional e (b) Visão Axial.

Além da informação obtida com o filtro de Sobel e vizinhança, deve-se garantir que serão pintados apenas voxels pertencentes ao pulmão a fim de, ao integrar esses voxels na segmentação, não sejam inclusos voxels externos ao pulmão, o que é garantido ao não pintar voxels com o valor máximo do *thresholding*. Após encontrar a borda, sabemos que dentro da borda estão os pulmões, brônquios e traqueia, e fora da borda estão as veias que devem posteriormente ser anexadas como pulmão e o background que deve ser segmentado sem cobrir as veias.

3.3 Remoção de background

O *background* (Figura 3.7b - vermelho) está conectado a veias (Figura 3.7b - branco) usando conectividade de face, isto é, conectividade tridimensional, pois as veias são es-

truturas de alta densidade que vêm de fora do pulmão para buscar oxigênio nos alvéolos pulmonares.

Figura 3.7: Segmentação de *background*. (a) Antes e (b) após segmentar o *background* em duas etapas.

Desejamos, nessa etapa, remover o *background* sem afetar as veias. Para isso, primeiro pintamos áreas externas a borda usando processamento em fatias axiais, isto é, usando conectividade planar, o que nos dá como resultado os elementos coloridos em vermelho na Figura 3.7b. Para isto, utilizamos sementes, em cada fatia axial, na área mais externa da figura e expandimos a pintura sobre todos os elementos sem pintar regiões internas a borda pulmonar, como pode ser visto no Algoritmo 10.

Algoritmo 10 Algoritmo para Remoção do Fundo

```

1: função FLOODFILLXY(x:inteiro , y:inteiro , z:inteiro, oldColor:inteiro , newColor:inteiro)
2:   color  $\leftarrow$  getColor(x, y, z)
3:   se color = oldColor então
4:     setColor(x, y, z, newColor)
5:     FloodFillXY(x - 1, y, z, oldColor, newColor)
6:     FloodFillXY(x + 1, y, z, oldColor, newColor)
7:     FloodFillXY(x, y - 1, z, oldColor, newColor)
8:     FloodFillXY(x, y + 1, z, oldColor, newColor)
9: função BACKGROUNDPARTIALREMOVAL(image:voxel***)
10:  para i = 0, i <= axialSlicesNumber(image), i ++ faça
11:    FloodFillXY(0, 0, i,  $\emptyset$ , BackgroundColor)

```

```

12: função BACKGROUNDHILARREGION(x:inteiro , y:inteiro , z:inteiro)
13:   color  $\leftarrow$  getColor(x, y, z)
14:   se color =  $\emptyset$  então
15:     setColor(x, y, z, BackgroundColor)
16:     FloodFillXY(x - 1, y, z,  $\emptyset$ , BackgroundColor)
17:     FloodFillXY(x + 1, y, z,  $\emptyset$ , BackgroundColor)
18:     FloodFillXY(x, y - 1, z,  $\emptyset$ , BackgroundColor)
19:     FloodFillXY(x, y + 1, z,  $\emptyset$ , BackgroundColor)
20:   se color = BackgroundColor então
21:     setColor(x, y, z, cutColor)
22:     BackgroundHilarRegion(x, y, z - 1)
23:     BackgroundHilarRegion(x, y, z + 1)
24:     BackgroundHilarRegion(x, y - 1, z)
25:     BackgroundHilarRegion(x, y + 1, z)

```

No procedimento, as veias não foram coloridas, permanecendo em sua cor original, Figura 3.7b - branco o que resulta uma segmentação parcial do *background*. Depois buscamos áreas de *background* encobertas próximo ao hilo pulmonar para completar a segmentação de *background*, esta região pode ser vista na Figura 3.7b - amarelo.

3.4 Segmentação de traqueia

A traqueia é encontrada por sua anatomia tubular. Algumas imagens estão rotacionadas no eixo caudal, eixo que liga a cabeça aos pés do paciente, então o algoritmo busca pela traqueia em ambas direções. As fatias axiais podem iniciar em qualquer ponto da boca até esôfago, o que significa diferentes anatomias nas primeiras fatias e além disso o formato da traqueia e esôfago são bem semelhantes então o algoritmo adapta-se as possíveis variações. A redução no *threshold* feito anteriormente ajuda a distinguir corretamente a traqueia do esôfago em totalidade das imagens (Figura 3.8). O Algoritmo 11 mostra em detalhes o procedimento para segmentação da traqueia.

O algoritmo baseia-se em, para cada fatia, em cada direção, buscar por uma traqueia que inicia nesta fatia na direção desejada. A traqueia é definida pelo algoritmo como um conjunto de pelo menos cinquenta fatias conexas com no máximo dois mil voxels de área, podendo nas fatias seguintes ter área maior desde que a área da região na fatia não cresça

em mais que três vezes o tamanho da fatia anterior, o que define uma estrutura tubular pouco rígida.

Figura 3.8: Segmentação da Traqueia.

Algoritmo 11 Segmentação da traqueia

```

1: função TRACHEAINITIALSLICESTRACH(image:voxel***)
2:   TracheaFound  $\leftarrow$  False;
3:   para  $i = 0, i < \text{axialSlicesNumber}(\text{image}), i++$  faça
4:     para todo  $\text{voxel} \in \text{slice}(i)$  faça
5:       se notPainted(voxel) & TracheaFound = False então
6:         TracheaFound  $\leftarrow$  TracheaSearch(voxel, 1)
7:     para todo  $\text{voxel} \in \text{slice}(\text{axialSlicesNumber}(\text{image}) - i)$  faça
8:       se notPainted(voxel) & TracheaFound = False então
9:         TracheaFound  $\leftarrow$  TracheaSearch(voxel, -1)
10: função TRACHEASEARCH(voxel, zDirection)
11:   TracheaElements[]  $\leftarrow$  {}
12:   ActualSeeds[]  $\leftarrow$  AxialConnectedElements(voxel)
13:   NextSeeds[]  $\leftarrow$  {}
14:   nSlices  $\leftarrow$  0
15:   enquanto true faça
16:     para todo  $\text{element} \in \text{ActualElements}$  faça
17:       queue(NextSeeds, Seed(element.x, element.y, element.z + zDirection))
18:     para todo  $\text{element} \in \text{NextSeeds}$  faça

```

```

19:         queue(NextElements, AxialConnectedElements(element))
20:     se (nSlices > 50 & size(NextElements) > 3*size(ActualElements)) | (nSlices <
    50 & size(NextElements) > 2000) então
21:         Break
22:     senão
23:         nSlices ++
24:         push(TracheaElements, ActualElements)
25:         ActualElements ← NextElements
26:         NextElements ← {}
27: se nSlices < 50 então
28:     retorna false
29: senão
30:     retorna true

```

No Algoritmo 11 temos duas funções, *TracheaInitialSliceSearch* e *TracheaSearch*, respectivamente, a função principal e auxiliar usadas no procedimento. A função principal recebe como parâmetro um volume que representa nossa imagem a ser segmentada.

Na Linha 2 definimos uma variável booleana para identificar se a traqueia já foi encontrada. O procedimento é iniciado na Linha 3 com um laço para que percorra todas as fatias axiais da imagem.

Nas Linhas 4 e 7 temos dois laços que executam tarefas semelhantes. Sendo o número i pertencente ao intervalo $0 \leq i < n$, onde n representa o número de fatias axiais, o primeiro laço na Linha 4 busca pela traqueia na fatia i , enquanto o segundo faz a busca na fatia $n - i$, ou seja, em fatias opostas.

Enquanto nas Linhas 4 e 7 são buscados todos os voxels pertencentes a fatias opostas, nas Linhas 5 e 8 é verificado se o voxel encontrado ainda não foi pintado e se a traqueia ainda não foi encontrada. Caso seja encontrado um voxel com essas características, nas Linhas 6 e 9 é buscado uma traqueia iniciando neste voxel, em sentidos contrários usando a função *TracheaSearch* definida na Linha 10.

Nas Linhas 11 a 13 são criados variáveis para armazenar os elementos já pintados presentes a traqueia em fatias anteriores, os elementos da fatia sendo pintada e os que serão pintados na fatia seguinte. Estes elementos são separados em variáveis diferentes para facilitar comparação de tamanho já que por ter anatomia tubular, a traqueia não deve sofrer alterações bruscas de tamanho entre fatias. Na Linha 14 é criada uma variável para

identificar o número de fatias já encontradas com anatomia compatível com a traqueia.

Na Linha 15 é iniciado um laço infinito que é terminado quando é identificado que a estrutura não é compatível com uma traqueia ou quando é detectado taxas de crescimento muito altas para uma traqueia o que ocorre quando a traqueia atinge a região hilar.

Nas Linhas 16 e 17 é buscado para cada elemento na fatia atual, quais elementos estão conexos a ele na fatia seguinte de acordo com o sentido a ser buscado que foi definido nas Linhas 6 e 9 da função principal. Feito isto temos, neste ponto do algoritmo, na variável *ActualSeeds* todos os elementos da fatia sendo pintada e na variável *NextSeeds*, todos os elementos conexos a *ActualSeeds* usando vizinhança de face.

Nas Linhas 18 e 19 são buscados elementos conexos a *NextSeeds* na fatia axial isto porque pode existir mais elementos conexos axialmente, mas que não são vizinhos de face de *ActualSeeds*.

Na Linha 20 é testado caso existam mais de cinquenta fatias já detectadas de traqueia, se o tamanho de *NextSeeds* é até três vezes superior a *ActualSeeds* ou, caso o número de fatias ser inferior e o tamanho de *NextSeeds* ser inferior a dois mil elementos.

Caso os tamanhos sejam superiores aos definidos na Linha 20 significa que a estrutura analisada teve grande crescimento, não sendo mais uma estrutura tubular, sendo o laço encerrado na Linha 21 e caso contrário, trata-se ainda de uma estrutura tubular sendo executado então as Linhas 23 a 26 que atualizam os valores das variáveis.

Caso seja uma traqueia, o laço é encerrado quando a traqueia for detectada até aproximadamente a região hilar, onde ela alcança o pulmão o que ocasiona grande taxa de crescimento deixando de ser uma estrutura tubular.

Nas Linhas 27 a 30 é retornado que foi encontrado uma traqueia caso o número de fatias identificadas seja superior a 50 fatias ou que não foi encontrado em caso contrário.

Após encontrada a traqueia, buscamos duas sementes. Uma a mais a esquerda possível e outra mais a direita possível da traqueia, que pertençam aos pulmões, que ainda não foram segmentados.

3.5 Divisão pulmonar

Neste ponto da execução, nós temos o *background* e traqueia segmentados e necessitamos segmentar os pulmões que podem estar conectados entre si. Para evitar vazamentos,

fazemos um corte no meio da imagem e a partir de duas sementes (esquerda e direita ao fim da traqueia), pintamos voxels conectados como pulmão esquerdo e direito, o que origina uma segmentação de pulmões parcial sem atravessar o corte (Figura 3.9a).

Figura 3.9: Divisão Pulmonar. (a) Corte inicial, (b) Complementação e (c) Otimização. Imagem meramente ilustrativa, pois com o auxílio do filtro de Sobel usado na detecção de bordas, esse método tem impacto menor sobre a imagem ilustrada considerando que o contorno é melhor detectado.

Algoritmo 12 Divisão Pulmonar

```

1: função FLOODFILL3D(x:inteiro , y:inteiro , z:inteiro, oldColor:inteiro , newColor:inteiro)
2:   color  $\leftarrow$  getColor(x, y, z)
3:   se color = oldColor então
4:     setColor(x, y, z, newColor)
5:     FloodFill3D(x - 1, y, z, oldColor, newColor)
6:     FloodFill3D(x + 1, y, z, oldColor, newColor)
7:     FloodFill3D(x, y - 1, z, oldColor, newColor)
8:     FloodFill3D(x, y + 1, z, oldColor, newColor)
9:     FloodFill3D(x, y, z - 1, oldColor, newColor)
10:    FloodFill3D(x, y, z + 1, oldColor, newColor)
11: função LUNGSEGMENTATION(image:voxel***, seedLeft:voxel, seedRight:voxel)
12:   depth  $\leftarrow$  axialSlicesNumber(image)
13:   height  $\leftarrow$  coronalSlicesNumber(image)
14:   width  $\leftarrow$  sagitalSlicesNumber(image)
15:   para k = 0, k < depth, k ++ faça
16:     para j = 0, j < height, j ++ faça
17:       se getColor(image[width/2][j][k]) =  $\emptyset$  então
18:         setColor(image[width/2][j][k], cutColor)

```

```

19:   FloodFill3D(getX(seedLeft), getY(seedLeft), getZ(seedLeft),  $\emptyset$ , lungLeft)
20:   FloodFill3D(getX(seedRight), getY(seedRight), getZ(seedRight),  $\emptyset$ , lungRight)
21:   para  $k = 0, k < \text{depth}, k++$  faça
22:       para  $j = 0, j < \text{coronalSlicesNumber}(\text{image}), j++$  faça
23:            $\text{colorLeft} \leftarrow \text{getColor}(\text{image}[\text{width}/2 - 1][j][k])$ 
24:            $\text{colorRight} \leftarrow \text{getColor}(\text{image}[\text{width}/2 + 1][j][k])$ 
25:           se  $\text{colorRight} = \emptyset$  então
26:               FloodFill3D( $\text{width}/2 + 1, j, k, \emptyset, \text{lungLeft}$ )
27:           se  $\text{colorLeft} = \emptyset$  então
28:               FloodFill3D( $\text{width}/2 - 1, j, k, \emptyset, \text{lungRight}$ )
29:   Move the remaining cuts by local search

```

No Algoritmo 12 temos duas funções *FloodFill3D* demonstrando como é definido o Algoritmo FloodFill em sua versão 3D e *LungSegmentation*, a função principal usada na segmentação de pulmões.

Na Linha 2 é obtida a cor do elemento passado como parâmetro, enquanto que na Linha 3 é verificado se a cor do elemento tem a cor que deve ser substituída. Caso afirmativo, é alterado a cor do elemento na Linha 4 e chamada a função recursivamente para todos os vizinhos de face nas Linhas 5 a 10.

Na Linha 11 é definida a nossa função para segmentação de pulmões, que recebe como parâmetros um volume que representa a imagem a ser segmentada e duas sementes, uma à esquerda e outra à direita da traqueia. Nas Linhas 12 a 14 são obtidos os tamanhos da imagem.

Nas Linhas 15 e 16 são criados laços para que em cada fatia da imagem seja feito um corte ao longo da altura no centro da imagem, isto é, onde $x = \text{width}/2$. Caso o elemento na posição de corte não tenha sido ainda pintado, o que é verificado na Linha 17, ele tem sua cor alterada na Linha 18.

Após feito o corte, nas Linhas 19 e 20, a partir das sementes iniciais, é preenchido todo o volume conexo aos voxels sementes. Nas Linhas 21 e 22 são analisadas, para cada elemento presente no corte feito, as cores dos elementos à esquerda e à direita do corte obtidos nas Linhas 23 e 24. Se um dos lados não estiver pintado, a cor do lado oposto é expandida, o que pode ser observado nas Linhas 25 a 28 e na Figura 3.9b.

Por último é feita uma busca local (RUSSELL; NORVIG, 2009), na Linha 29, onde são analisados os números de voxels à esquerda e à direita do corte, movendo o corte para o

lado com menor número de voxels, até que o corte seja feito em um ponto ótimo local, minimizando a área do corte. Deste modo, nós temos uma separação de pulmões mais precisa (Figura 3.9c).

Vários cortes independentes podem ser movidos em direções diferentes em cada imagem porque os pulmões podem estar conectados em muitos pontos diferentes. Mais detalhes podem ser vistos no Algoritmo 12. Este algoritmo poderia ser melhorado com uso de outras técnicas, porém mesmo para o caso ilustrado na Figura 3.9, este procedimento não é mais necessário pois ao utilizar o filtro de Sobel, esta imagem teve sua borda corretamente segmentada, não sendo mais utilizado este procedimento, mas mantemos o algoritmo para casos que possam não vir a ser bem segmentados apenas com a segmentação de borda.

3.6 Vasos sanguíneos

Agora que tudo já foi segmentado com exceção dos voxels que representam veias, fazemos o processamento de veias pintando todos os voxels que tocam na borda pulmonar evitando incluir áreas danificadas por patologias ao pulmão, isto é, grandes massas (Figura 3.10a).

Figura 3.10: Segmentação de Vasos Sanguíneos. (a) antes e (b) depois de verificar vizinhança dos voxels usando algoritmo k -NN.

Agora que o pulmão está segmentado entre pulmão esquerdo e direito, e veias envoltas por pulmão, fazemos uso do algoritmo k -NN (do Inglês, *k Nearest Neighbor*) o qual é usado para classificação de elementos.

No k -NN, um elemento a ser classificado pergunta a classe dos k elementos mais próximos obtendo sua classificação pelo voto da maioria dos vizinhos, como pode ser visto no Algoritmo 13. Assim, para cada voxel pertencente a classe "veia", pergunta-se a seus 26 vizinhos, usando a vizinhança completa exibida na Figura 1.2d, a qual classe os vizinhos pertencem, descobrindo então se o voxel de veia a ser rotulado está envolto por

elementos de pulmão esquerdo ou direito, classificando-o como veia esquerda ou direita, respectivamente.

Algoritmo 13 Separação de veias

```

1: função VESSELSSEGMENTATION(borderSeeds:voxel*)
2:   para todo voxel  $\in$  borderSeeds faça
3:     numberLeft  $\leftarrow$  0
4:     numberRight  $\leftarrow$  0
5:     x  $\leftarrow$  getX(voxel)
6:     y  $\leftarrow$  getY(voxel)
7:     z  $\leftarrow$  getZ(voxel)
8:     para k = -1, k  $\leq$  1, k ++ faça
9:       para j = -1, j  $\leq$  1, j ++ faça
10:        para i = -1, i  $\leq$  1, i ++ faça
11:          se getColor(x + i, y + j, z + k) = lungLeft então
12:            numberLeft ++
13:          se getColor(x + i, y + j, z + k) = lungRight então
14:            numberRight ++
15:        setColor(x, y, z,  $\emptyset$ )
16:      se numberLeft > numberRight então
17:        FloodFill3D(x, y, z,  $\emptyset$ , vesselLeft)
18:      senão
19:        FloodFill3D(x, y, z,  $\emptyset$ , vesselRight)

```

No Algoritmo 13, na Linha 1, temos a definição da função responsável pela segmentação de veias e recebe como parâmetro um conjunto de voxels que fazem parte da borda pulmonar. Esses elementos são usados como sementes no procedimento e na Linha seguinte é criado um laço para que seja executado o algoritmo para cada semente.

Nas Linhas 3 a 7 são iniciados variáveis com a posição do voxel a ser analisado e quantidade de elementos que pertencem a pulmão esquerdo e direito e são vizinhos do voxel analisado usando a vizinhança completa de um voxel.

Nas Linhas 8 a 14 é verificado para cada vizinho presente na vizinhança completa, se ele é um voxel já previamente classificado como pulmão esquerdo ou direito para que seja incrementado seu respectivo contador.

Na Linha 15 é apagada a cor do voxel semente e caso a predominância de vizinhos

sejam de pulmão esquerdo, esse elemento é usado como semente para ser pintado uma estrutura com cor de veias de pulmão esquerdo ou caso contrário será pintado uma estrutura com cor reservada para estrutura de veia de pulmão direito.

Ao final de algoritmo, todos os elementos que tocam na fronteira pulmonar e não foram ainda segmentados são pintados com cor de veias e de acordo com a predominância de vizinhos, eles recebem a cor identificando se são veias do pulmão esquerdo ou direito. Isto funciona pois nesse momento todas as outras estruturas já foram segmentadas.

Neste caso, para cada elemento de veia presente na Figura 3.10a, o algoritmo pergunta aos vizinhos, os quais envolvem as veias e já se encontram rotulados como pulmão esquerdo ou direito, se maioria deles pertencem ao pulmão esquerdo ou direito a fim de classificar as veias como veias pertencentes ao pulmão esquerdo ou direito. Vários outros algoritmos poderiam funcionar nesta etapa. O k -NN foi escolhido por simplicidade de implementação.

3.7 Segmentação final

Repintando a borda como traqueia, pulmão esquerdo ou direito, nós temos a segmentação da imagem (Figura 3.11a) obtida com o auxílio do algoritmo k -NN (RUSSELL; NORVIG, 2009).

Figura 3.11: Resultado final. (a) Axial (b) Tridimensional.

Exportando para o padrão de referência usado por LOLA11 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) que considera a traqueia como *background* e veias como pulmão, temos o resultado final que pode ser visto na Figura 3.11b.

4 Segmentação de Lóbulos Usando o SALSA

Para realizar segmentação de lóbulos a primeira etapa deve ser a detecção das fissuras que dividem o pulmão em lóbulos. Porém, as fissuras em geral têm apenas parte de seu volume visível devido ao fato de serem muito finas, gerando descontinuidades na imagem original. Assim, buscamos desenvolver um método que não dependa desta informação, mas que possa obter segmentação precisa dos lóbulos pulmonares.

O método proposto é baseado na informação da segmentação de veias, que podem ser separadas pela distância entre as subárvores formadas, como ilustrado na Figura 4.1 que demonstra o resultado alcançado até o momento pelo nosso método.

Figura 4.1: Objetivo da segmentação de veias, subárvores de veias formadas por lóbulo.

4.1 Complementação de veias

Na Seção 3.6, apresentamos um algoritmo rápido para segmentação de veias. Porém, podemos conseguir mais informação de um outro método, baseado na análise dos autovalores da matriz hessiana.

Em matemática, a matriz hessiana de uma função de n variáveis é uma matriz quadrada com valores das derivadas parciais de segunda ordem da função. Por isto, esta matriz descreve a curvatura local da função usada.

A matriz hessiana foi desenvolvida no século XIX pelo alemão Ludwig Otto Hesse, razão porque mais tarde James Joseph Sylvester lhe deu este nome. O próprio Hesse, ao contrário, usava o termo "determinantes funcionais". A matriz Hessiana pode ser calculada segundo a Equação 4.1.

$$H[f(x_1, x_2, x_3, \dots, x_n)] = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (4.1)$$

Sendo nossa função de interesse, a função que define a nossa imagem tomográfica, que pode ser definida como $f(x, y, z)$, sendo (x, y, z) a coordenada para um elemento da imagem (voxel). Assim, a função obtém a intensidade de cor de um elemento da imagem. Devemos, então, aplicar a hessiana sobre nossa função f a fim de poder estudar a curvatura local de nossa imagem, o que pode ser feito analisando os autovalores de nossa matriz resultante, $H[f(x, y, z)]$.

Para encontrar os autovalores, devemos buscar se o determinante de nossa matriz H subtraído de um escalar λ multiplicado pela matriz identidade I , é igual a zero. Caso positivo, teremos um conjunto de valores escalares λ , que chamamos de autovalores da matriz. Veja a aplicação na Equação 4.2.

$$\begin{aligned}
\det(H[f(x, y, z)] - \lambda I) &= \det \left(\begin{bmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \\
&= \det \begin{bmatrix} f_{xx} - \lambda & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} - \lambda & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} - \lambda \end{bmatrix} \\
&= (f_{xx} - \lambda) * (f_{yy} - \lambda) * (f_{zz} - \lambda) + f_{xy} * f_{yz} * f_{zx} \\
&\quad + f_{xz} * f_{yx} * f_{zy} - f_{xz} * (f_{yy} - \lambda) * f_{zx} \\
&\quad - f_{yz} * f_{zy} * (f_{xx} - \lambda) - (f_{zz} - \lambda) * f_{xy} * f_{yx}
\end{aligned} \tag{4.2}$$

Para solucionarmos o sistema da Equação 4.2, sabemos que derivadas parciais em mesmas variáveis são iguais, devido a nossa função ter variáveis independentes, o que nos leva a Equação 4.3 após manipularmos os valores.

$$\begin{aligned}
\det(H - \lambda I) &= (f_{xx} - \lambda)(f_{yy} - \lambda)(f_{zz} - \lambda) + 2(f_{xy}f_{yz}f_{xz}) \\
&\quad - f_{xz}^2(f_{yy} - \lambda) - f_{yz}^2(f_{xx} - \lambda) - f_{xy}^2(f_{zz} - \lambda) \\
&= \lambda^3 - \lambda^2(f_{xx} + f_{yy} + f_{zz}) + \lambda(f_{xx}f_{yy} + f_{xx}f_{zz} + f_{yy}f_{zz} - f_{xy}^2 - f_{xz}^2 - f_{yz}^2) \\
&\quad - 2(f_{xy}f_{yz}f_{xz}) + f_{xz}^2f_{yy} + f_{yz}^2f_{xx} + f_{xy}^2f_{zz} - f_{xx}f_{yy}f_{zz}
\end{aligned} \tag{4.3}$$

A Equação 4.3, pode ser reescrita na forma $\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0$, onde a_2, a_1 e a_0 , estão definidos nas Equações 4.4, 4.5 e 4.6, respectivamente.

$$a_2 = f_{xx} + f_{yy} + f_{zz} \tag{4.4}$$

$$a_1 = f_{xx}f_{yy} + f_{xx}f_{zz} + f_{yy}f_{zz} - f_{xy}^2 - f_{xz}^2 - f_{yz}^2 \tag{4.5}$$

$$a_0 = -2(f_{xy}f_{yz}f_{xz}) + f_{xz}^2f_{yy} + f_{yz}^2f_{xx} + f_{xy}^2f_{zz} - f_{xx}f_{yy}f_{zz} \tag{4.6}$$

Para calcular derivadas parciais de nossa função f , podemos usar o método das diferenças finitas, onde podemos definir as derivadas parciais de primeira ordem como exibido nas Equações 4.7, 4.8 e 4.9 que são conhecidas por, respectivamente, fórmula das diferenças progressivas, regressivas e centradas.

Para uso deste método, devemos escolher uma constante h que representa o desloca-

mento entre elementos da imagem. Assim, para um valor pequeno de h , o erro do cálculo, $o(h)$, é esperado que seja um erro pequeno então normalmente é atribuído o valor de uma unidade a constante h .

$$f_x = \frac{f(x+h) - f(x)}{h} + o(h) \quad (4.7)$$

$$f_x = \frac{f(x) - f(x-h)}{h} + o(h) \quad (4.8)$$

$$f_x = \frac{f(x+h) - f(x-h)}{2h} + o(h^2) \quad (4.9)$$

Para nossa função f , usando a fórmula das diferenças centradas e deslocamento h igual a 1, menor valor possível em nossa função imagem, temos como obter analiticamente as derivadas parciais f_x , f_y e f_z , conforme demonstrado nas Equações 4.10, 4.11 e 4.12.

$$f_x \simeq \frac{f(x+1, y, z) - f(x-1, y, z)}{2} \quad (4.10)$$

$$f_y \simeq \frac{f(x, y+1, z) - f(x, y-1, z)}{2} \quad (4.11)$$

$$f_z \simeq \frac{f(x, y, z+1) - f(x, y, z-1)}{2} \quad (4.12)$$

De forma similar, podendo ser aplicado mais de uma vez o método das diferenças finitas, pode ser calculado derivadas de segunda ordem, como exibido nas Equações 4.13, 4.14 e 4.15. Aplicando a mesma idéia, podemos calcular também as derivadas mistas como exibido nas Equações 4.16, 4.17 e 4.18.

$$\begin{aligned} f_{xx} &\simeq \frac{f_x(x+1, y, z) - f_x(x-1, y, z)}{2} \\ &\simeq \frac{\frac{f(x+1+1, y, z) - f(x+1-1, y, z)}{2} - \frac{f(x-1+1, y, z) - f(x-1-1, y, z)}{2}}{2} \\ &\simeq \frac{f(x+2, y, z) - 2f(x, y, z) + f(x-2, y, z)}{4} \end{aligned} \quad (4.13)$$

$$\begin{aligned} f_{yy} &\simeq \frac{f_y(x, y+1, z) - f_y(x, y-1, z)}{2} \\ &\simeq \frac{\frac{f(x, y+1+1, z) - f(x, y+1-1, z)}{2} - \frac{f(x, y-1+1, z) - f(x, y-1-1, z)}{2}}{2} \\ &\simeq \frac{f(x, y+2, z) - 2f(x, y, z) + f(x, y-2, z)}{4} \end{aligned} \quad (4.14)$$

$$\begin{aligned}
f_{zz} &\simeq \frac{f_z(x, y, z+1) - f_z(x, y, z-1)}{2} \\
&\simeq \frac{\frac{f(x, y, z+1+1) - f(x, y, z+1-1)}{2} - \frac{f(x, y, z-1+1) - f(x, y, z-1-1)}{2}}{2} \\
&\simeq \frac{f(x, y, z+2) - 2f(x, y, z) + f(x, y, z-2)}{4}
\end{aligned} \tag{4.15}$$

$$\begin{aligned}
f_{xy} &\simeq \frac{f_x(x, y+1, z) - f_x(x, y-1, z)}{2} \\
&\simeq \frac{\frac{f(x+1, y+1, z) - f(x-1, y+1, z)}{2} - \frac{f(x+1, y-1, z) - f(x-1, y-1, z)}{2}}{2} \\
&\simeq \frac{f(x+1, y+1, z) - f(x-1, y+1, z) - f(x+1, y-1, z) + f(x-1, y-1, z)}{4}
\end{aligned} \tag{4.16}$$

$$\begin{aligned}
f_{xz} &\simeq \frac{f_x(x, y, z+1) - f_x(x, y, z-1)}{2} \\
&\simeq \frac{\frac{f(x+1, y, z+1) - f(x-1, y, z+1)}{2} - \frac{f(x+1, y, z-1) - f(x-1, y, z-1)}{2}}{2} \\
&\simeq \frac{f(x+1, y, z+1) - f(x-1, y, z+1) - f(x+1, y, z-1) + f(x-1, y, z-1)}{4}
\end{aligned} \tag{4.17}$$

$$\begin{aligned}
f_{yz} &\simeq \frac{f_y(x, y, z+1) - f_y(x, y, z-1)}{2} \\
&\simeq \frac{\frac{f(x, y+1, z+1) - f(x, y-1, z+1)}{2} - \frac{f(x, y+1, z-1) - f(x, y-1, z-1)}{2}}{2} \\
&\simeq \frac{f(x, y+1, z+1) - f(x, y-1, z+1) - f(x, y+1, z-1) + f(x, y-1, z-1)}{4}
\end{aligned} \tag{4.18}$$

Conforme visto na Equação 4.3, para a resolução, é gerado um polinômio de grau 3 sobre a variável λ , assim, a solução terá três valores de λ , λ_1 , λ_2 e λ_3 que podem ser encontrados usando a formulação de Cardano-Tartaglia que encontra os coeficientes para uma equação do terceiro grau da forma $x^3 + a_2x^2 + a_1x + a_0 = 0$, a qual é semelhante a função encontrada na Equação 4.3 devido também ser uma equação do terceiro grau com o primeiro coeficiente a_1 com valor 1.

A partir de uma mesma imagem f , obtém-se então para cada ponto (x, y, z) , três valores de λ , ou podemos dizer de forma semelhante, que a função $f(x, y, z)$ gera três novas funções $\lambda_1(x, y, z)$, $\lambda_2(x, y, z)$ e $\lambda_3(x, y, z)$, que são também imagens tridimensionais.

Autovalores são importantes em aplicações de processamento de imagens. Existem vários métodos disponíveis e em nosso trabalho devido ao uso de imagens tomográficas, optamos por usar o Insight Toolkit (ITK) (INSIGHT..., Acessado em Setembro, 2016) para

gerar as imagens de autovalores para nosso objetivo. O ITK é uma suíte de ferramentas para análise em imagens que é open-source, independente de plataforma e implementa algoritmos para registrar e segmentar dados multidimensionais.

Assumindo que os autovalores estejam ordenados tal que $\lambda_1 < \lambda_2 < \lambda_3$, as imagens λ_1, λ_2 e λ_3 , têm características diferentes, sendo λ_1 , útil para analisar grandes variações de cor, como fronteiras entre órgãos. Na Figura 4.2, podemos ver a imagem gerada pelos valores de λ_1 , no eixo axial, sagital e coronal. A imagem, para melhor impressão teve sua escala de cores invertida, isto é, a escala de branco até preto foi invertida.

Figura 4.2: Imagens geradas pelos valores de lambda. Usando lambda 1, os planos de corte (a) axial, (b) sagital e (c) coronal, usando lambda 2, os planos de corte (d) axial, (e) sagital e (f) coronal e com lambda 3, os planos de corte (g) axial, (h) sagital e (i) coronal.

Já as imagens geradas pelos valores de λ_2 são úteis para analisar locais de pouca

variação de cor, permitindo identificar estruturas como veias. Na Figura 4.2, podemos ver a imagem gerada pelos valores de λ_2 , no eixo axial, sagital e coronal. As imagens geradas por λ_2 são difíceis de analisar devido ocorrer pouco contraste. Para melhoria da visibilidade foi efetuado um processo de thresholding para gerar a Figura 4.2, a fim de tornar preto estruturas na faixa de valor para veias e branco todo o resto.

Por último as imagens geradas por λ_3 destacam membranas estreitas, como por exemplo a membrana que separa os lóbulos pulmonares. Na Figura 4.2, podemos ver a imagem gerada pelos valores de λ_3 , no eixo axial, sagital e coronal. Para gerar a imagem com os valores de λ_3 , foi feita uma redução na escala para que a partir de determinado valor, a cor fosse sempre preta, um achatamento na escala para tornar zonas de baixa intensidade mais evidentes. Aqui podemos notar um plano que corta os pulmões aproximadamente no centro da imagem, a Pleura Visceral.

Dito isto, usamos a imagem obtida por λ_2 para buscar estruturas que complementem as veias obtidas com o algoritmo da Seção 3.6 e obtivemos sucesso, sendo um método complementar. Porém, a imagem acompanha muitas pequenas estruturas irrelevantes e/ou desconexas além da informação desejada, como pode ser melhor visto na Figura 4.3.

4.2 Segmentação de veias em lóbulos

Baseado na segmentação de veias mostrado na Figura 4.3b, foi implementada a remoção de estruturas irrelevantes, descartando estruturas que ao ser adicionadas, têm volume pequeno, indicando ser vasos sanguíneos estreitos irrelevantes como mostrado na Figura 4.3c.

Após o descarte de pequenas estruturas, foi gerado uma imagem 2D no plano sagital sobrepondo as veias e calculado a distância de cada ponto para a veia mais próxima, representado na Figura 4.4a e 4.4b. Em preto, veias sobrepostas no plano sagital, em tons de cinza, do mais escuro ao mais claro, respectivamente, de próximo a mais distante de veias.

Apesar de estruturas pequenas terem sido removidas, ainda restam algumas, como pode ser visto pelos pontos soltos na Figura 4.4c. A fim de removê-las, para tornar mais clara a separação entre lóbulos, foi efetuado um processo de erosão na sobreposição de veias a fim de remover locais de baixa importância, como pode ser visto nas Figuras 4.4c e 4.4d.

(a)

(b)

(c)

Figura 4.3: Segmentação de veias, visão de cima. (a) Veias obtidas com o algoritmo obtido na Seção 3.6 (b) Veias obtidas usando Autovalores λ_2 sobre a matriz hessiana e (c) Subtração das estruturas 3D que geraram as imagens anteriores

Figura 4.4: Mapeamento de distâncias, com e sem uso de erosão sobre as veias antes de mapear a distância. (a) e (c) Pulmão esquerdo, (b) e (d) Pulmão direito, respectivamente sem erosão e com uso de erosão antes de mapear as distâncias. Em destaque as duas maiores regiões conexas de veias para o pulmão esquerdo e as três maiores regiões para o pulmão direito.

Nota-se na Figura 4.4c e 4.4d que foram mantidas as áreas de veia mais espessas e acentuada a distância entre veias de lóbulos distintos. Usando a imagem gerada, buscamos pelas maiores regiões conexas de veias, assumindo que o pulmão não foi operado para remoção de lóbulos, buscamos pelas duas maiores regiões conexas no pulmão esquerdo e três maiores regiões conexas no pulmão direito.

Usando as sementes de veias obtidas da análise de grupamentos de veias, semeamos em todas as fatias sagitais a fim de segmentar os grupamentos de veias de forma tri-

Figura 4.5: Segmentação de veias baseado em sementes. Lóbulo superior em vermelho e bege, respectivamente, do pulmão direito e esquerdo. Lóbulo inferior em verde e marrom, respectivamente, pertencentes ao pulmão direito e esquerdo. Em azul o lóbulo médio e em cinza, veias não segmentadas.

dimensional, buscando elementos de veias conexos a estas sementes. O resultado deste procedimento pode ser visto na Figura 4.5.

4.3 Watershed

Existem algumas variações do algoritmo de *watershed*, sendo a mais antiga introduzida por S. Beucher e C. Lantuéjoul em 1979 (BEUCHER; LANTUÉJOUL, 1979), conhecida como o algoritmo de *watershed* por inundação. O conceito por trás do algoritmo é processar a imagem com uma dimensão extra referente a intensidade dos tons de cinza indicando a profundidade do elemento como em um mapa topográfico. Neste caso, o algoritmo simula o despejar de água sobre alguns pontos e modela o comportamento da água acumulando-se nos vales da imagem até que a imagem esteja completamente inundada.

Para evitar que águas vindas de fontes diferentes misturem-se, são criadas barreiras. O resultado da execução do algoritmo é a informação de onde estão as barreiras e uma segmentação da imagem onde regiões tendem a separar-se nos locais mais claros da imagem. Um dos algoritmos usando essa estratégia foi introduzido por F. Meyer na década de 90 (BEUCHER; MEYER, 1992).

Para que o algoritmo seja executado, é preciso que seja fornecido uma imagem em escala de cinza a ser transformada pelo comportamento do algoritmo onde os elementos

Figura 4.6: Estágios do algoritmo de watershed (a) imagem original, (b) visão topográfica com pontos semente escolhidos e (c) imundação inicial, (d) imundação completa. Fonte: Digital Image Processing (GONZALEZ; WOODS, 2002)

mais claros da imagem devem ser parte da fronteira que buscamos detectar pela execução do algoritmo, isto é, a separação entre lóbulos. O problema enfrentado para construção desta imagem é que a pleura visceral, a qual buscamos identificar é muito delicada e tem apenas parte de seu volume visível o que no algoritmo de watershed causaria vazamentos entre regiões, comprometendo a qualidade da segmentação.

A fim de obter uma imagem de qualidade a usar no *watershed*, combinamos a informação de obtida pelos autovalores da matriz hessiana λ_3 , que podem conter descontinuidades na pleura visceral, com o mapeamento de distâncias de veias mostrado na seção 4.2 que

Algoritmo 14 Algoritmo Watershed de Meyer

1. Selecionar um conjunto de pixels onde a inundação deve iniciar são escolhidos e marcados com diferentes rótulos.
 2. Inserir os vizinhos de cada área marcada em uma fila de prioridade onde o nível de cinza correspondente ao nível de prioridade.
 3. Extrair o pixel com a menor prioridade da fila e rotulado. Inserir na fila todos os vizinhos não rotulados que estão fora da fila de prioridade.
 4. Refazer a etapa anterior até a fila estar vazia.
-

não sofre de descontinuidades, porém é menos preciso que os dados obtidos por λ_3 . Na Figura 4.7, podemos ver a montagem da imagem usada em nosso watershed.

Figura 4.7: Imagem de custo e resultado final da segmentação obtido pela aplicação do algoritmo *watershed* sobre a matriz de custo obtida combinando a imagem gerada por λ_3 e o mapeamento de distância entre veias.

5 Discussão e Resultados

Os resultados atuais em LOLA11 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) usando segmentação automática são descritos abaixo separados em duas seções. Resultados da segmentação pulmonar e da segmentação de lóbulos pulmonares. Os resultados do SALSA foram obtidos usando um notebook I5-3337U CPU 1.8Ghz, 4Gb Ram DDR3-1600 e HD SSD.

5.1 Resultados da segmentação pulmonar

Os resultados foram obtidos a partir da competição *Lobe and Lung Analysis 2011* (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) a qual disponibiliza a imagem de referência para a segmentação e o resultado obtido por nosso algoritmo para algumas fatias sagitais para que possamos estudar o comportamento do método.

Ao todo são fornecidos imagens de comparação usando cinco imagens, Lola11-01, Lola11-22, Lola11-25, Lola11-37 e Lola11-45. Para duas primeiras imagens, não notamos diferença entre o padrão de referência e o resultado da segmentação usando o SALSA. Exibimos na Figura 5.1 as imagens fornecidas por LOLA11 (RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) onde ocorreram maiores diferenças para compararmos o desempenho de nosso método para os casos onde ocorreram maiores diferença em relação ao padrão de referência. Abaixo, nossos resultados.

Na Tabela 2, nas colunas temos, respectivamente, o objeto (obj) analisado que pode ter como valores LL e RL (do Inglês, *Left Lung* e *Right Lung*, respectivamente), a média (mean), o desvio padrão (std), o mínimo (min), o primeiro quartil chamado também de quartil inferior (q1), o segundo quartil também conhecido como mediana (median), o terceiro quartil chamado de quartil superior (q3) e, por último, o valor máximo (max).

Quinze grupos submeteram seus métodos de segmentação pulmonar para o LOLA11, destes, temos resultados de pontuação melhor que sete deles e pior que outros sete, abaixo

Figura 5.1: Segmentação de pulmões SALSA, a esquerda, o padrão de referência e a direita, o resultado obtido pelo SALSA. (a-b) Lola11-25, (c-d) Lola11-37 e (e-f) Lola11-45

descrevemos mais detalhadamente alguns grupos e seus resultados a fim de comparar não apenas a pontuação obtida como também o tempo de processamento.

Tabela 2: Sobreposição para cada pulmão, para 54 imagens em LOLA11.

obj	mean	std	min	Q1	median	Q3	max
LL	0.959	0.108	0.213	0.968	0.985	0.989	0.994
RL	0.957	0.135	0.015	0.978	0.986	0.991	0.995
score	0.958						

Tabela 3: Consumo de tempo (segundos), para 55 imagens em LOLA11

obj.	mean	std	min	Q1	median	Q3	max
lung	47.13	38.62	10.32	31.03	37.17	52.45	248.66

Tabela 4: Consumo de Tempo/Voxel (microsegundos), para 55 imagens em LOLA11

obj.	mean	std	min	Q1	median	Q3	max
lung	0,435	0,393	0,156	0,413	0,359	0,354	3,040

- Weinheimer et al (WEINHEIMER et al., 2011) apresenta pontuação de 0.964 (sem fechamento morfológico) and 0.97 (com fechamento morfológico). Tempo médio de execução em um PC (Intel Xeon CPU, 2.83 GHz, 4GB RAM) por caso foi 7.3 minutos sem fechamento morfológico e 11.4 minutos com fechamento morfológico.
- Lassen et al (LASSEN et al., 2011) apresenta pontuação de 0.973 usando um computador quad-core padrão com Windows, 8 GB de RAM e consumiu cerca de 1 minuto por caso em média. Tempos variaram entre 0.5 e 3 minutos.
- Pinho et al (PINHO; DELMON; VANDEMEULEBROUCKE, 2011) apresenta pontuação de 0.948. Não descreveu o tempo de execução.
- Korn et al (KORN et al., 2011), apresenta pontuação de 0.949. O tempo médio foi 12.6 minutos usando um notebook Dell com Intel Core2Duo CPU T9600 a 2.8Ghz com 4GB de RAM.
- Sun et al (SUN; BAUER; BEICHEL, 2011), apresenta pontuação de 0.949. Em média, cerca de 6 minutos foi necessário, mas não foi descrito as características do computador usado.
- Gu et al (GU; PU, 2011), apresenta pontuação de 0.939. Não descreveu o tempo de execução.
- Hosseini et al (HOSSEINI-ASL; ZURADA; EL-BAZ, 2011), apresenta pontuação de 0.965. Não descreveu o tempo de execução.
- Soliman et al (SOLIMAN et al., 2015) obteve o melhor resultado por um algoritmo automático de segmentação, 0.98. Não descreveu o tempo de execução.

- O melhor resultado da segmentação foi alcançado por segmentação totalmente manual executada pelo grupo Human seguindo o protocolo fornecido pela competição, o que resultou na pontuação de 0.984.

O SALSA teve melhor pontuação que Gu et al (GU; PU, 2011), Pinho et al (PINHO; DELMON; VANDEMEULEBROUCKE, 2011), Korn et al (KORN et al., 2011), Sun et al (SUN; BAUER; BEICHEL, 2011) e outros três grupos e pior pontuação que Weinheimer et al (WEINHEIMER et al., 2011), Lassen et al (LASSEN et al., 2011), Hosseini et al (HOSSEINI-ASL; ZURADA; EL-BAZ, 2011) e outros quatro grupos, porém nosso resultado é apenas 0,022 pontos abaixo do melhor método de segmentação automático.

O tempo de execução do SALSA foi calculado usando um computador com clock inferior que os outros descritos acima e usando um computador com menor clock, o SALSA necessita apenas de 47 segundos em média e tempo variou entre 0.16 e 4:14 minutos. Comparando o tempo médio, conseguimos executar 9 vezes mais rápido que Weinheimer et al (WEINHEIMER et al., 2011), 20 por cento mais rápido que Lassen et al (LASSEN et al., 2011), 16 vezes mais rápido que Korn et al (KORN et al., 2011), 7 vezes mais rápido que Sun et al (SUN; BAUER; BEICHEL, 2011). Outros grupos analisados não descreveram seus tempos de execução.

5.2 Resultados da segmentação de lóbulos pulmonares

Ainda não temos resultados de acurácia para a segmentação de lóbulos pulmonares pois ainda não submetemos nossos dados para análise de acurácia por LOLA11 (RIKXO-ORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014) devido estarmos em fase de testes e melhorias no método. Na Figura 5.2, mostramos nossos resultados qualitativos para algumas imagens onde obtivemos boa segmentação e casos patológicos ainda não abordados pelo nosso método.

Podemos observar nas Figuras 5.2b e 5.2c vazamentos no procedimento de watershed enquanto para casos como em 5.2d, o algoritmo falhou ao detectar a membrana que separa os lóbulos pulmonares. O método está sendo melhorado, mas já mostra resultados iniciais.

Figura 5.2: Segmentação de lóbulos pulmonares. (a) Lola11-01, (b) Lola11-02, (c) Lola11-03, (d) Lola11-05

6 Considerações Finais

O trabalho desenvolvido foi continuado a partir do trabalho de conclusão de curso de mesmo autor, onde na época, o método proposto não era capaz de segmentar pulmões com patologias graves, gerando resultados inaceitáveis e devido erros de programação, alguns casos sequer tinham resultado por mal funcionamento do algoritmo. Todo o método de segmentação foi revisto e refeito, alcançando boa pontuação na área de segmentação pulmonar.

Além disto, foi proposto um método simples e com um diferencial em relação aos métodos estudados para segmentação do pulmão em lóbulos, problema de difícil solução com melhores resultados alcançados por Lassen et al. (LASSEN et al., 2011). O algoritmo de Lassen busca identificar sementes iniciais para os lóbulos com base na árvore traquéia bronquial, ao invés disso nosso método demonstra ser possível obter estas sementes através apenas da análise de veias, não sendo necessário identificar a árvore traquéia bronquial.

A matriz de custo do *watershed* criada por Lassen é obtida, em parte, pelo mapeamento de distâncias de cada voxel para a veia mais próxima, porém temos conhecimento de que a membrana que separa o pulmão em lóbulos é aproximada por uma reta no plano sagital e calculando as distâncias com base na sobreposição de veias no plano sagital, obtivemos imagem mais nítida sobre o local da membrana de forma mais simples.

Devido a membrana que separa o pulmão em lóbulos, a pleura visceral, ser muito fina, muitas vezes não é identificada totalmente o que afeta bastante a segmentação dos lóbulos pulmonares, razão pela qual existem vários métodos para segmentação de pulmões e apenas dois para a segmentação de lóbulos em LOLA11(RIKXOORT; GINNEKEN; KERKSTRA, Acessado em Novembro, 2014). Sendo um deles fruto da tese de doutorado de Lassen.

Em trabalhos futuros sugere-se a união do método proposto com algoritmos de malha para evitar vazamentos na fronteira detectada melhorando a segmentação. Sugere-se também a análise dos agrupamentos de veias para automaticamente identificar pacientes que foram submetidos a procedimentos de lobotomia, isto é, remoção de lóbulos pulmonares.

Referências

- BERG, M. de et al. *Computing the Maximum Overlap of Two Convex Polygons Under Translations*. Department of Computer Science, Utrecht University, 1996. 613–628 p. Disponível em: <<https://books.google.com.br/books?id=BW1trgEACAAJ>>.
- BEUCHER, S.; LANTUÉJOUL, C. *Use of Watersheds in Contour Detection*. set. 1979. International Workshop on image. Disponível em: <<http://cmm.ensmp.fr/~beucher/publi/watershed.pdf>>.
- BEUCHER, S.; MEYER, F. *The morphological approach to segmentation: the watershed transformation*. : Marcel Dekker AG, 1992. 433–433 p.
- COSTA, A.; CARVALHO, B. M. Salsa - A simple automatic lung segmentation algorithm. In: PARDO, A.; KITTLER, J. (Ed.). *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 20th Iberoamerican Congress, CIARP 2015, Montevideo, Uruguay, November 9-12, 2015, Proceedings*. : Springer International Publishing, 2015. p. 501–508. ISBN 978-3-319-25751-8.
- DEFINIENS XD. Acessado em Setembro, 2016. [Www.definiens.com](http://www.definiens.com).
- DOEL, T.; GAVAGHAN, D. J.; GRAU, V. *Review of automatic pulmonary lobe segmentation methods from CT*. Oct 2014.
- GIMP. Acessado em Maio, 2015. [Www.gimp.org](http://www.gimp.org).
- GONZALEZ, R.; WOODS, R. *Digital Image Processing*. Second ed. : Prentice Hall, 2002.
- GRAND Challenges in Biomedical Image Analysis. Acessado em Setembro, 2016. Grand-challenge.org.
- GU, S.; PU, J. Automatic lung segmentation improvement using a registration framework. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.
- HOSSEINI-ASL, E.; ZURADA, J. M.; EL-BAZ, A. Lung segmentation using incremental sparse nmf. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.
- INSIGHT Toolkit. Acessado em Setembro, 2016. [Www.itk.org](http://www.itk.org).
- ITK-SNAP. Acessado em Novembro, 2014. [Www.itksnap.org](http://www.itksnap.org).
- KORN, R. et al. Description of a fully automatic lung segmentation algorithm based on the cognition network technology. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.

- LASSEN, B. et al. Lung and lung lobe segmentation methods at fraunhofer mevis. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.
- PINHO, R.; DELMON, V.; VANDEMEULEBROUCKE, J. Keuhkot: A method for lung segmentation. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.
- RIKXOORT, E. van; GINNEKEN, B. van; KERKSTRA, S. *Lobe and Lung Analysis 2011*. Acessado em Novembro, 2014. Disponível em: <<http://www.lola11.com/>>.
- RUDYANTO, R. D. et al. *VESsel SEgmentation in the Lung 2012*. Acessado em Maio, 2016. Disponível em: <<http://vessel12.grand-challenge.org/>>.
- RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Third ed. : Prentice Hall, 2009.
- SKETCHUP. Acessado em Maio, 2015. [Www.sketchup.com](http://www.sketchup.com).
- SOLIMAN, A. et al. Segmentation of pathological lungs from CT chest images. In: . 2015.
- SUN, S.; BAUER, C.; BEICHEL, R. Robust active shape model based lung segmentation in CT scans. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.
- WEINHEIMER, O. et al. Automatic lung segmentation in MDCT images. In: *Fourth International Workshop on Pulmonary Image Analysis (in conjunction with MICCAI 2011)*. 2011.