EDWARD FLÓREZ PACHECO

**QUANTIFICAÇÃO DA DINÂMICA DE ESTRUTURAS EM IMAGENS DE
MEDICINA NUCLEAR NA MODALIDADE PET**

**SÃO PAULO
2012**

EDWARD FLÓREZ PACHECO

**QUANTIFICAÇÃO DA DINÂMICA DE ESTRUTURAS EM IMAGENS DE
MEDICINA NUCLEAR NA MODALIDADE PET**

**Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para a obtenção do título de MESTRE, pelo
Programa de Pós-Graduação em
Engenharia Elétrica, Área de Sistemas
Eletrônicos, opção Engenharia Biomédica.**

Orientador Prof. Dr. Sérgio Shiguemi Furuie

**São Paulo
2012**

Este exemplar foi revisado e alterado em relação à versão original sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 09 de Março de 2012.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Flórez Pacheco, Edward

**Quantificação da dinâmica de estruturas em imagens de medicina nuclear na modalidade PET / E. Flórez Pacheco. – ed.rev. -- São Paulo, 2012.
199 p.**

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

**1. Medicina nuclear 2. Imageamento (Bioengenharia)
I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II. t.**

DEDICATÓRIA

A Deus,

Porque me deu aquele poder de sempre me sentir capaz e não ter medo a conseguir novos logros e objetivos.

Porque apesar dos logros conseguidos até agora, não me afastou das coisas que verdadeiramente queria conseguir, mesmo tendo uma posição ganhada com muito esforço me deu sempre muita vontade e segurança para poder começar o caminho que sempre sonhei.

Por colocar pessoas tão especiais e essenciais em todo momento da minha vida, que me ajudaram a superar os obstáculos e me motivaram a seguir pelo caminho certo.

Porque sempre me colocou no lugar certo, no tempo certo e com as pessoas certas. Ele foi testemunho em todo momento da vontade, esforço e perseverança que tive ao longo do meu caminho profissional e pessoal.

Porque sempre preencheu minha vida de coisas maravilhosas e não deixou de me proteger e me conduzir pelo caminho verdadeiro, aumentou sempre minha fé e me fez melhor filho, melhor amigo e melhor estudante.

*Aos meus adorados pais: **Emperatriz e Edgard***, duas pessoas muito especiais e únicas, exemplos de pais e amigos, quem estiveram presentes de coração e espírito em cada um dos passos na minha vida, perto ou longe de casa. Transmitiram-me sempre a confiança, segurança e a força necessária para seguir meu rumo. Ensinaram-me que posso errar, mas posso corrigir os erros; que posso parar um pouco, mas nunca me deter.

A vocês que se esforçaram sempre para me dar tudo do melhor, muito mais do que realmente precisei. Desejo que saibam que vocês significam o melhor do melhor neste mundo. Muito obrigado por me acompanharem com compreensão, e com um incomparável e incondicional amor, acreditando sempre nos meus planos e objetivos.

Ao meu irmão **Roosevelt** e a minha irmã **Milagros**, pelo carinho sincero, apoio constante e por sempre serem aqueles parceiros ao longe da minha existência. Sempre me deram aquele espaço tão especial em suas vidas, acompanhando de perto meu desenvolvimento e comemorando como suas cada uma de minhas conquistas.

A minha adorada **Susana**. Você é aquela pessoa essencial e única na minha vida, que Deus me permitiu conhecer e partilhar dia-a-dia seja de perto o mesmo na distância; por isso: Sei que podemos estar juntos em todo instante mesmo sem estar no mesmo lugar, sei que podemos nos sentir só com o pensamento, sei que posso sentir muita saudade de você, mas também sei que posso te amar muito mais. Tenho certeza que nada é maior, forte e verdadeiro que nosso amor.

A toda minha família, por toda a força, apoio, respaldo e carinho. Pela alegria e esforço de cada um de vocês para sempre estarem atentos em todos os meus passos neste longo caminho de aperfeiçoamento profissional. Por sempre ter essas palavras de motivação e confiança que fizeram muito bem.

AGRADECIMENTOS ESPECIAIS

Ao meu orientador e amigo **Prof. Dr. Sérgio Shiguemi Furuie**, pela confiança, pela compreensão dos meus anseios e por me dar o incentivo necessário para atingir os meus objetivos.

Porque em cada conversa, nunca deixou de expressar sua opinião, sempre com a intenção de me ensinar o caminho certo, dentro do mais correto.

Porque aprendi muito da sua experiência e ética profissional.

Pela habilidade na arte de ensinar, e explicar os assuntos mais complicados com uma clareza e simplicidade extraordinária.

Por sempre ter um tempo para me atender mesmo com sua agenda de muitas atividades, me dar suporte, e realizar as revisões detalhadas que foram muito importantes no desenvolvimento, execução e conclusão deste trabalho.

AGRADECIMENTOS

À **Escola Politécnica da Universidade de São Paulo – USP**, na pessoa de seu Diretor, Prof. Dr. José Roberto Cardoso.

Aos **Professores do Programa de Pós-Graduação em Engenharia Biomédica**, pela contribuição na minha formação acadêmica e científica.

Ao **Serviço de Informática do Instituto do Coração (InCor) do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo (HC-FMUSP)**, pela ajuda e facilidades no fornecimento das imagens médicas para a elaboração deste projeto.

Ao **Ministério de Relações Exteriores Perú – Brasil** e ao **CNPq** pelo apoio financeiro.

A meus amigos da graduação e a vida toda: **Cinthia, Gina, José, Alexander, Diego, Walter, Alex, Mario, Omar, Richard, Miguel e Wilder**; pela parceria, e porque a pesar de ter tomado caminhos diferentes, sempre achamos um instante para nos reunir e construir mais momentos juntos. Ficaram sempre no meu coração.

A meus amigos e minhas amigas das casas onde morei desde que cheguei ao Brasil: **Melissa, Ana Paula, Janaina, Gibrán, Rafael, Erick, Karen, Anderson e Késsia**, que me receberam de braços abertos e me deram a mão amiga desde o primeiro dia que me conheceram e apesar de estar longe de casa, sempre conseguimos construir uma linda família com muito respeito e igualdade.

À **Nilza e Aparecido**, que me acolheram sempre como um membro da sua família, muitíssimo obrigada pelo carinho e consideração.

Aos meus amigos de turma do Programa de Pós Graduação: **Mônica, Murielle, João Eduardo, André, Matheus, Fernando, Diego** e colegas do Programa da Pós-Graduação, **Jimmy, Renato, Wilder, Leonardo, Heiner, Diego, Marcelo, Diana, Lina, Tayla, Rosangela, Catarina e Aline** pela partilha e convívio, pelas lindas festas surpresas, pelos gestos de companheirismo, carinho e trabalho em equipe.

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

*"El futuro tiene muchos nombres. Para los débiles es lo
inalcanzable. Para los temerosos, lo desconocido. Para los
valientes es la oportunidad."
Victor Hugo*

*"El único secreto real del éxito es el entusiasmo."
Walter Chrysler*

*"Entre maior é a prova, mais glorioso é o triunfo."
Anônimo*

ÍNDICE

LISTA DE FIGURAS	13
LISTA DE TABELAS	17
LISTA DE ABREVIATURAS E SIGLAS	20
RESUMO	21
ABSTRACT.....	23
 1. INTRODUÇÃO	 25
1.1 Considerações Gerais	25
1.2 Objetivos	27
 2. REVISÃO DA LITERATURA	 28
2.1 PET como técnica de Imagem Médica	28
2.1.1 Evolução histórica da técnica PET	28
2.1.2 Processos Físicos na técnica PET	31
2.1.3 Comparação do PET com respeito às outras modalidades de imagens médicas.....	36
2.1.4 Principais aplicações do PET	38
2.3 Presente e futuro do PET	41
2.3.1 Situação atual no Brasil	41
2.3.2 Perspectivas no futuro	43
2.4 Conectividade Fuzzy de Imagens Digitais	44
2.4.1 Definições Iniciais da Teoria Fuzzy.....	46
2.4.2 Espaço Digital Fuzzy.....	46
2.4.3 Cena Fuzzy, Cena Binária e Cena de Pertinência.....	47
2.4.4 Adjacência e Afinidade Fuzzy.....	47
2.4.5 Conectividade Fuzzy e Objeto Fuzzy.....	48
 3. METODOLOGIA	 50
3.1 Conceitos Iniciais.....	50
3.2 Diagrama de Blocos	51
3.3 Aquisição/Simulação das imagens 3D.....	52
3.4 Filtragem de Imagens.....	56
3.5 Segmentação de Imagens.....	62
3.6 Avaliação dos resultados.....	66

3.7	Quantificação dos Volumes.....	67
4.	IMPLEMENTAÇÃO	68
4.1	Aquisição / Simulação de Imagens Médicas 3D.....	69
4.1.1	Ventrículo Esquerdo simulado.....	69
4.1.2	<i>Phantoms</i> simulados com o NCAT-4D.....	71
4.1.3	Imagens PET reais.....	72
4.2	Adição de Ruído quântico tipo Poisson.....	73
4.2.1	Avaliação do Ruído Poisson.....	74
4.3	Pre-processamento: Filtragem das imagens 3D.....	77
4.4	Segmentação de estruturas de interesse nas imagens 3D.....	81
4.4.1	Curvas Características de Operação do Receptor (ROC).....	82
4.5	Pós filtragem através de morfologia numérica.....	83
4.6	Avaliação dos Resultados.....	84
4.7	Quantificação dos Volumes.....	86
5.	RESULTADOS	89
5.1	VE simulado: Imagens Padrões.....	89
5.2	VE simulado: Imagens com Ruído.....	90
5.3	VE simulado: Imagens Filtradas.....	92
5.4	<i>Phantom</i> do VE gerado pelo NCAT-4D.....	93
5.5	Imagens PET reais.....	94
6.	DISCUSSÃO	96
6.1	Importância e contribuições da pesquisa	96
7.	CONCLUSÕES	98
8.	DESAFIOS E PROJETO FUTURO	100
	REFERÊNCIAS	103
	APÊNDICE A	109
A.1	– PARÂMETROS DE AQUISIÇÃO DO VE SIMULADO.....	109

APÊNDICE B.....	110
B.1 – PARÂMETROS DO <i>PHANTOM</i> DO VE ATRAVÉS DO NCAT-4D.....	110
APÊNDICE C.....	112
C.1 – DADOS EXAME PET REAL.....	112
APÊNDICE D.....	116
D.1 – GOODNESS OF FIT TEST.....	116
D.2 – TABELA TEORICA DA DISTRIBUIÇÃO QUI-QUADRADO.....	128
APÊNDICE E.....	129
E.1 – CARACTERISTICAS DE OPERAÇÃO DO RECEPTOR (ROC).....	129
APÊNDICE F.....	130
F.1 – AVALIAÇÃO DA SEGMENTAÇÃO DAS IMAGENS 3D.....	130
APÊNDICE G.....	133
E.1 – COMPARATIVO DOS TEMPOS DE SEGMENTAÇÃO.....	133
APÊNDICE H.....	134
G.1 – LISTAGEM DE PROGRAMAS.....	134
APÊNDICE I.....	199
H.1 – PUBLICAÇÕES RESULTATES DESTE TRABALHO.....	199

LISTA DE FIGURAS

FIGURA 1	-	Evolução histórica da qualidade da imagem PET desde sua origem (Imagem Siemens http://www.cerebromente.org.br/).	30
FIGURA 2	-	O mais moderno aparelho lançado pela GE: Optima PET/CT 560 (Imagem GE Healthcare, http://www.gehealthcare.com/).	30
FIGURA 3	-	Esquema da emissão e aniquilação do pósitron. A modalidade PET é baseada neste processo físico (Imagem adaptada de http://www.hardmob.com.br/threads/438088Aniquila%C3%A7%C3%A3odemat%C3%A9riaentimateriapoder%C3%A1-criar-laser-de-Raios-Gama/).	33
FIGURA 4	-	Esquema de aniquilação com representação do efeito da não colinearidade [López, 2008].	33
FIGURA 5	-	Dispersão Compton no Objeto emissor [López, 2008].	34
FIGURA 6	-	Os cristais convertem a radiação gama, emitidos do paciente, a fótons de luz, e os tubos fotomultiplicadores converte e amplifica os fótons a sinais elétricos. Estes sinais elétricos então são processados pelo computador para finalmente gerar a imagem dessa região. O aparelho então pode ser movido, e com o processo repetido, se obtêm uma série de imagens (fatias) do corpo sobre a região de interesse (Imagem adaptada de http://health.howstuffworks.com/medicine/modern/nuclear-medicine1.htm).	35
FIGURA 7	-	Comparativo do volume dos procedimentos de diagnóstico de imagem por modalidade (Imagem adaptada de http://maxbley.typepad.com/2007/09/if-im-perfectly.html).	42
FIGURA 8	-	Exemplo de segmentação do círculo na imagem simulada para demonstrar a efetividade da Conectividade Fuzzy em relação à um método mais simples, como por exemplo o Region Growing - (a) Imagem simulada original, (b) Imagem com ruído Gaussiano, (c) Relação de afinidade de cada spel, (d) Círculo segmentado por crescimento de região (percebe-se que uma parte dele foi perdida), (e) Círculo segmentado por Conectividade Fuzzy [Udupa; Saha, 2003].	45

FIGURA 9	-	Diagrama de blocos geral do projeto.	51
FIGURA 10	-	(A) Vista anterior do <i>phantom</i> NCAT (3D), mostrando a segmentação detalhada de órgãos. (B) Imagem 3D do modelo cardíaco do <i>phantom</i> NCAT, com a descendente anterior esquerda (do inglês left anterior descending - LAD) e circunflexa esquerda (do inglês left circumflex - LCX), ramos da artéria coronária sobreposta à superfície exterior do modelo do coração. (C) Imagem 3D do modelo do ventrículo esquerdo do <i>phantom</i> NCAT mostrando as superfícies internas e externas, a LAD e LCX, os ramos da árvore arterial coronária, uma placa aterosclerótica e um simulado de um infarto do miocárdio. Fonte das imagens do paper: Normal and Pathological NCAT Image and Phantom Data Based on Physiologically Realistic Left Ventricle Finite Element Models [Veress et al., 2006].	54
FIGURA 11	-	Phantom tri-dimensional do Ventrículo Esquerdo gerado em MatLab e reconstruído utilizando ferramentas do ImageJ.	55
FIGURA 12	-	Representação do ruído Poisson: Em (a) se mostra a imagem original do VE de entrada, já em (b) temos a imagem com o ruído Poisson incorporado (Typical PET image - http://courses.washington.edu/).	57
FIGURA 13	-	Ilustração do procedimento para redução do ruído Poisson utilizando a transformada de Anscombe (AT), o filtro de Wiener pontual e a Transformada Inversa de Anscombe (IAT).	61
FIGURA 14	-	Ilustração geométrica dos fatores utilizados no processo de Avaliação [Udupa et al., 2002] .	67
FIGURA 15	-	Estrutura de processamento de imagens PET (3D) implementado como parte do projeto.	68
FIGURA 16	-	Simulação simples do Ventrículo Esquerdo do coração. (Imagem das partes do coração adaptada de http://campus.fortunecity.com/).	71
FIGURA 17	-	Phantom simulado com o uso do NCAT-4D. (a) Conjunto de fatias 2D das projeções dos órgão pelo eixo z, (b) Vista frontal do volume gerado com todas as fatias usando o ImageJ.	72

FIGURA 18	- Phantom simulado com o uso do NCAT-4D. (a) fatias 2D das projeções do ventrículo esquerdo dentro da caixa torácica, (b) volume gerado com ajuda do ImageJ.	72
FIGURA 19	- Exame PET real. (a) fatias 2D das projeções das cavidades do miocárdio, (b) volume gerado por meio de ferramentas de reconstrução do Image J.	73
FIGURA 20	- (a) fatia 2D de uma secção do ventrículo esquerdo dentro da caixa torácica, (b) fatia 2D com ruído Poisson.	74
FIGURA 21	- (a) Imagem de uma fatia do VE simulado, (b) fatia do VE simulado com ruído Poisson incorporado e (c) resultados qualitativos do filtro da mediana sobre o ruído Poisson com três tamanhos de janelas diferente. Foram alterados os níveis de contraste e brilho das imagens mostradas, mudando a escala de intensidades para melhor visualização.	77
FIGURA 22	- (a) Imagem de uma fatia do phantom gerado através do NCAT-4D, (b) fatia com ruído Poisson incorporado e (c) resultados qualitativos do filtro da mediana sobre o ruído Poisson com três tamanhos de janelas diferente.	78
FIGURA 23	- Resultados qualitativos do filtro da Gaussiana ponderada sobre o ruído Poisson do VE simulado, usando diferentes valores de sigma. Foram alterados os níveis de contraste e brilho das imagens mostradas, mudando a escala de intensidades para melhor visualização.	79
FIGURA 24	- Resultados qualitativos do filtro da Gaussiana ponderada sobre o ruído Poisson do phantom gerado por meio do NCAT-4D, usando diferentes valores de sigma.	79
FIGURA 25	- Perfil (linha central horizontal) de (a) Imagem com ruído Poisson e (b) Imagem filtrada.	80
FIGURA 26	- Phantom do ventrículo esquerdo simulado com o uso do NCAT-4D. (a) Conjunto de fatias 2D das projeções do VE junto com as estruturas da caixa torácica, (b) Fatias 2D obtidas depois do processo de Segmentação, utilizando o algoritmo Conectividade Fuzzy com Pesos Dinâmicos.	82
FIGURA 27	- Diagrama de Blocos do Projeto de Doutorado.	102
FIGURA 28	- Principais parâmetros do phantom ncat_LV60bpm_log. Arquivo fornecido pelo Serviço de Informatica do InCor - HCFMUSP.	111

FIGURA 29 - Volume PET N° 36 usando ferramentas de reconstrução 3D do ImageJ. 115

LISTA DE TABELAS

Tabela 1	-	Listado dos radiofármacos comumente empregados na modalidade PET [Pozzo, 2005].	37
Tabela 2	-	Comparação das principais características entre as modalidades de Imagens Médicas existentes [López, 2008].	38
Tabela 3	-	Principais parâmetros considerados no desenvolvimento do Ventrículo Esquerdo Simulado.	70
Tabela 4	-	Quadro comparativo dos resultados dos algoritmos de filtragem vs. segmentação para o volume do VE simulado.	85
Tabela 5	-	Quadro comparativo dos resultados dos algoritmos de filtragem vs. segmentação para o phantom NCAT-4D.	86
Tabela 6	-	Quadro comparativo dos resultados de quantificação (expressados em pixels) do volume PET N° 36.	87
Tabela 7	-	Parâmetros de Avaliação no processo de Segmentação das Imagens Padrões para o grupo de Homens (10 volumes).	89
Tabela 8	-	Parâmetros de Avaliação no processo de Segmentação das Imagens Padrões para o grupo de Mulheres (10 volumes).	90
Tabela 9	-	Parâmetros de Avaliação no processo de Segmentação das Imagens Padrões para o grupo de Crianças (10 volumes).	90
Tabela 10	-	Parâmetros de Avaliação no processo de Segmentação das Imagens com Ruído Poisson para o grupo de Homens (10 volumes).	91
Tabela 11	-	Parâmetros de Avaliação no processo de Segmentação das Imagens com Ruído Poisson para o grupo de Mulheres (10 volumes).	91
Tabela 12	-	Parâmetros de Avaliação no processo de Segmentação das Imagens com Ruído Poisson para o grupo de Crianças (10 volumes).	91
Tabela 13	-	Parâmetros de Avaliação no processo de Segmentação após a etapa de filtragem através do Filtro Anscombe/Wiener para o grupo de Homens (10 volumes).	92
Tabela 14	-	Parâmetros de Avaliação no processo de Segmentação após a etapa de filtragem através do Filtro Anscombe/Wiener para o grupo de Mulheres (10 volumes).	92

Tabela 15	-	Parâmetros de Avaliação no processo de Segmentação após a etapa de filtragem através do Filtro Anscombe/Wiener para o grupo de Crianças (10 volumes).	93
Tabela 16	-	Parâmetros de Avaliação no processo de Segmentação das imagens Padrões, com Ruído e Filtradas (usando o Filtro de Anscombe/Wiener) do <i>Phantom</i> ncat_LV60bpm_log que representa o VE do coração.	93
Tabela 17	-	Resultados da quantificação do volume PET N° 33.	94
Tabela 18	-	Resultados da quantificação do volume PET N° 34.	94
Tabela 19	-	Resultados da quantificação do volume PET N° 35.	95
Tabela 20	-	Resultados da quantificação do volume PET N° 36.	95
Tabela 21	-	Parâmetros dos Volumes do VE utilizados no sequência de processamento, foram divididos em três grupos (Homens, Mulheres e Crianças) com dez diferentes volumes cada.	109
Tabela 22	-	Reporte do exame PET: Observam-se os tempos de aquisição dos 36 volumes PET, assim como o consumo do radiofármaco em cada período de tempo.	113
Tabela 23	-	Goodness of Fit test, Ruído Poisson – $\lambda = 5$.	116
Tabela 24	-	Goodness of Fit test, Ruído Poisson – $\lambda = 10$.	117
Tabela 25	-	Goodness of Fit test, Ruído Poisson – $\lambda = 20$.	119
Tabela 26	-	Goodness of Fit test, Ruído Poisson – $\lambda = 30$.	121
Tabela 27	-	Goodness of Fit test, Ruído Poisson – $\lambda = 40$.	123
Tabela 28	-	Goodness of Fit test, Ruído Poisson – $\lambda = 50$.	125
Tabela 29	-	Goodness of Fit test, Ruído Gaussiano – $\lambda = 50$.	127
Tabela 30	-	Tabela teórica da Distribuição Qui-Quadrado.	128
Tabela 31	-	Resultados da segmentação do VE simulado usando o algoritmo Genérico, considerando diferentes pesos de w_1 e w_2 .	129
Tabela 32	-	Resultados da segmentação do VE simulado usando o algoritmo LIFO, considerando diferentes pesos de w_1 e w_2 .	129
Tabela 33	-	Resultados da segmentação do VE simulado usando o algoritmo KTetaFOEMS, considerando dif. pesos de w_1 e w_2 .	129
Tabela 34	-	Resultados parciais do processo de segmentação das Imagens Padrões do primeiro volume do grupo de Homens (Caso_H01).	130
Tabela 35	-	Resultados parciais do processo de segmentação das Imagens com Ruído Poisson do primeiro volume do grupo de Homens (Caso_H01).	131

Tabela 36	-	Resultados parciais do processo de segmentação das Imagens Filtradas do primeiro volume do grupo de Homens (Caso_H01).	132
Tabela 37	-	Comparativo das médias dos tempos utilizados no processo de segmentação do VE simulado para os quatro algoritmos implementados.	133

LISTA DE ABREVIATURAS E SIGLAS

AT	Anscombe Transform (Transformada de Anscombe)
PET	Positron Emission Tomography (Tomografia por Emissão de Pósitrons)
CT	Computed Tomography (Tomografia Computadorizada)
CAD	Coronary Artery Disease (Doença arterial coronariana)
SPECT	Single Photon Emission Computed Tomography
2D	2 dimensões ou bidimensional - uma imagem plana.
3D	3 dimensões ou tridimensional - uma imagem composta.
API	Application Programming Interface (Interface de Programação de Aplicativos)
DP	Desvio padrão.
DICOM	Digital Imaging and Communications in Medicine (Comunicação de imagens digitais em Medicina).
FC	Fuzzy Connectedness (Conectividade Fuzzy).
FP	Falso Positivo.
GFC	Generic Fuzzy Connectedness (conectividade nebulosa genérica).
IAT	Inverse Anscombe Transform (Transformada de Anscombe Inversa).
IFT	Image Foresting Transform (Transformada imagem-floresta).
InCor	Instituto do Coração do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo.
JNI	Java Native Interface (Interface entre códigos Java e códigos nativos).
LIFO	Last input – first output (estrutura de administração de dados).
MRI	Magnetic resonance imaging (Ressonância Magnética)
Pixel	Picture element (elemento de imagem).
RMSE	Normalized Root Mean Square Error (raiz normalizada do erro médio quadrático)
ROC	Receiver Operating Characteristic (Características de Operação do Receptor)
ROI	Region of interest (Região de interesse).
Spel	Spacial element (elemento espacial, pode ser um píxel ou um voxel 3D).
SO	Sistema Operacional.
UML	Unified Modeling Language (Linguagem Unificada de Modelagem).
Voxel	Volume element (elemento de volume).
VE	Ventrículo Esquerdo.
VP	Verdadeiro Positivo.

Flórez Pacheco E. Quantificação da Dinâmica de Estruturas em Imagens de Medicina Nuclear na modalidade PET [Dissertação]. São Paulo: Escola Politécnica da Universidade de São Paulo, Faculdade da Elétrica, Área de Engenharia Biomédica.

RESUMO

A presença que tem hoje a Medicina Nuclear como modalidade de obtenção de imagens médicas é muito importante e um dos principais procedimentos utilizados hoje nos centros de saúde, tendo como grande vantagem a capacidade de conseguir analisar o comportamento metabólico do paciente, fazendo possíveis diagnósticos precoces. Este projeto está baseado em imagens médicas obtidas através da modalidade PET (Positron Emission Tomography) a qual está tendo uma crescente difusão e aceitação. Para isso, temos desenvolvido uma estrutura integral de processamento de imagens tridimensionais PET, a qual está constituída por etapas consecutivas que se iniciam na obtenção das imagens padrões (gold standard), sendo utilizados volumes simulados ou *phantoms* do Ventrículo Esquerdo do Coração criadas como parte do projeto, assim como geradas a partir do software NCAT-4D. A seguir, nos volumes simulados, é introduzido ruído quântico tipo Poisson que é o ruído característico das imagens PET e feita uma análise que busca certificar que o ruído utilizado corresponde efetivamente ao ruído Poisson. Em sequência é executada a parte de pré-processamento, utilizando para este fim, um conjunto de filtros tais como o filtro da mediana, o filtro da Gaussiana ponderada e o filtro que mistura os conceitos da Transformada de Anscombe e o filtro pontual de Wiener. Posteriormente é aplicada a etapa de segmentação que é considerada a parte central da sequência de processamento. O processo de segmentação é baseado na teoria de Conectividade Fuzzy e para isso temos implementado quatro diferentes abordagens: Algoritmo Genérico, Algoritmo LIFO, Algoritmo kTetaFOEMS e o Algoritmo utilizando Pesos Dinâmicos. Sendo que os três primeiros algoritmos utilizam pesos específicos selecionados pelo usuário, foi preciso efetuar uma análise para determinar os melhores pesos de segmentação que se reflitam numa segmentação mais eficiente. Finalmente,

para terminar a estrutura de processamento, um procedimento de avaliação foi utilizado como métrica para obter quantitativamente três parâmetros (Verdadeiro Positivo, Falso Positivo e Máxima Distância) que permitiram conhecer o nível de eficiência e precisão de nosso processo e do projeto em geral. Constatamos que os algoritmos implementados (filtros e algoritmos de segmentação) são bastante robustos e atingem ótimos resultados chegando-se a obter, para o caso do volume do Ventrículo Esquerdo simulado, taxas de VP e FP no ordem de $98.49 \pm 0.27\%$ e $2,19 \pm 0.19\%$, respectivamente. Com o conjunto de procedimentos e escolhas feitas ao longo da estrutura de processamento, encerramos o projeto com a análise de um grupo de volumes produto de um exame PET real, obtendo a quantificação destes volumes.

PALAVRAS-CHAVE: Processamento de imagens tri-dimensionais, Segmentação de imagens tri-dimensionais, Medicina Nuclear, Tomografia por Emissão de Pósitrons (PET), Conectividade Fuzzy, Quantificação dos volumes, *Phantoms*, Imagens PET reais.

Flórez Pacheco E. Quantification of Dynamic Structures in Nuclear Medicine Images in the PET modality [Dissertation]. São Paulo: Polytechnic School of São Paulo University, Electrical School, Biomedical Engineering Area.

ABSTRACT

The usefulness of Nuclear medicine nowadays as a modality to obtain medical images is very important, and it has turned into one of the main procedures utilized in Health Care Centers. Its great advantage is to analyze the metabolic behavior of the patient, by allowing early diagnosis. This project is based on medical images obtained by the PET modality (Positron Emission Tomography), which has won wide acceptance. Thus, we have developed an integral framework for processing Nuclear Medicine three-dimensional images of the PET modality, which is composed of consecutive steps that start with the generation of standard images (gold standard) by using simulated images or phantoms of the Left Ventricular Heart that were generated in this project, such as the ones obtained from the NCAT-4D software. Then Poisson quantum noise is introduced into the whole volume to simulate the characteristic noises in PET images and an analysis is performed in order to certify that the utilized noise is the Poisson noise effectively. Subsequently, the pre-processing is executed by using specific filters, such as the median filter, the weighted Gaussian filter, and the filter that joins the concepts of Anscombe Transformation and the Wiener filter. Then the segmentation, which is considered the most important and central part of the whole process, is implemented. The segmentation process is based on the Fuzzy Connectedness theory and for that purpose four different approaches were implemented: Generic algorithm, LIFO algorithm, kTetaFOEMS algorithm, and Dynamic Weight algorithm. Since the first three algorithms used specific weights that were selected by the user, an extra analysis was performed to determine the best segmentation constants that would reflect an accurate segmentation. Finally, at the end of the processing structure, an assessment procedure was used as a measurement tool to quantify some parameters that determined the level of efficiency and precision of our

process and project. We have verified that the implemented algorithms (filters and segmentation algorithms) are fairly robust and achieve optimal results, assist to obtain, in the case of the Left Ventricular simulated, TP and FP rates in the order of $98.49 \pm 0.27\%$ and $2.19 \pm 0.19\%$, respectively. With the set of procedures and choices made along of the processing structure, the project was concluded with the analysis of a volumes group from a real PET exam, obtaining the quantification of the volumes.

KEYWORDS: Digital Image Processing, Segmentation of tri-dimensional images, Nuclear Medicine, Positron Emission Tomography (PET), Fuzzy Connectedness, Volume quantification, Phantoms, Real PET images.

Capítulo 1

INTRODUÇÃO

1.1 Considerações Gerais

Na atualidade existem algumas modalidades de imagens médicas, entendendo-se por imagem médica o conjunto de técnicas, métodos ou procedimentos através dos quais é possível obter imagens de um organismo vivo, ou parte dele [Bushberg et al., 2002]. As quatro principais modalidades utilizadas hoje na medicina são: Ultrassom, Ressonância Magnética, Raios X / Tomografia Computadorizada e Medicina Nuclear. Todas elas têm como precursor os Raios X, o qual existe há um século de antiguidade [Caetano, 2002].

Cada uma destas modalidades possui um princípio físico diferenciado que lhe permite obter diferentes tipos de informações dos pacientes. Desta forma, cada modalidade apresenta vantagens e limitações, o que faz que cada uma delas seja adequada para aplicações pontuais. As imagens médicas geradas através destas modalidades conformam dois grupos bem diferenciados: imagens projetivas e imagens tomográficas. O primeiro grupo é integrado por um conjunto de imagens bidimensionais, as quais são apropriadas para inspeções e como ferramentas de diagnóstico, entre elas, as radiografias (raios X) e as cintilografias (fazendo uso de raios gama). Já no outro grupo encontramos as imagens tomográficas, cuja principal diferença é a capacidade de reconstruir imagens tridimensionais.

Atualmente, é crescente a aceitação e utilização da Medicina Nuclear onde encontramos a Tomografia por emissão de pósitrons (PET em diante, acrônimo do inglês *Positron Emission*

Tomography). O PET é uma modalidade de imagem médica que dá a possibilidade de reconhecer alterações num órgão, função e/ou estrutura específica de seres humanos e/ou de animais, com peculiar sensibilidade e especificidade na detecção de doenças. Consegue também demonstrar precocemente desordens, transtornos ou alterações bioquímicas, mesmo onde não existe (ainda) uma anormalidade estrutural evidente, permitindo um diagnóstico mais precoce e com maior possibilidade de acerto [De Almeida et al., 2002; Robilotta, 2006].

O PET possibilitou à Medicina Nuclear observar a distribuição espacial e temporal de marcadores fisiológicos e funcionais, permitindo avaliação quantitativa e não-invasiva de processos fisiológicos. Estes marcadores podem indicar desde o funcionamento anômalo de órgãos até a bioquímica no metabolismo celular. Sua principal aplicação consiste na investigação de tumores e metástases, que naturalmente têm aumentado o consumo de glicose, devido ao maior nível metabólico. Se o estudo é realizado em vários intervalos sucessivos de tempo, obtém-se uma distribuição temporal de imagens do marcador, possibilitando estudos dinâmicos [Tinois, 2005].

O futuro do PET é auspicioso, ajudado com o desenvolvimento de novos radiofármacos e materiais detectores, técnicas de reconstrução mais eficientes, entre outros avanços. No entanto, ainda há inúmeros mecanismos fisiológicos que são desconhecidos ou pouco esclarecidos, e o PET é uma importantíssima ferramenta para investigar hipóteses e modelos de fenômenos biológicos, inclusive na efetividade das diferentes terapias. Neste processo, há uma carência muito grande em relação a técnicas e ferramentas computacionais que permitam avaliar e estudar a dinâmica de processos a partir de imagens tridimensionais de PET.

1.2 Objetivos do Projeto

O objetivo principal do projeto é a pesquisa e desenvolvimento de métodos para a análise da dinâmica de processos fisiológicos baseado em PET. Especificamente, será investigada a análise de volumes de estruturas de interesse ao longo do tempo.

A relevância deste trabalho consiste no desenvolvimento de métodos computacionais que permitirão avaliar uma estrutura completa de processamento de imagens 3D de Medicina Nuclear na modalidade PET, com o propósito de determinar quais são os procedimentos que melhor desempenho oferecem dentro do projeto.

O projeto terá como foco o estudo do ventrículo esquerdo do coração, no entanto, como ferramenta de aplicação não ficará limitada a esta região.

Capítulo 2

REVISÃO DA LITERATURA

Neste capítulo, uma introdução ao PET como uma modalidade de Imagem Médica, será apresentada. Uma visão geral do tema, assim como quais são seus principais problemas, desafios, aplicações atuais e suas perspectivas no futuro serão abordadas. Conhecer estes aspectos permitirá compreender melhor os resultados mostrados na secção cinco. Para facilitar o entendimento dos estudos relacionados a este tema de investigação, a revisão de literatura foi dividida nos seguintes tópicos:

2.1 PET como modalidade de Imagem Médica

Esta primeira etapa, não constitui um trabalho de investigação propriamente dito, mas sim pretende expor um conjunto de idéias e dados relevantes que buscam dar o sustento necessário a nossa pesquisa. O conteúdo vertido neste capítulo é o resultado da revisão de diversas fontes de informação. Com ele pretende-se unicamente pôr em contexto as idéias e conceitos que nos seguintes capítulos serão apresentados, analisados e discutidos.

2.1.1 Evolução histórica da modalidade PET

Com princípios claramente diferenciados daquelas ferramentas não invasivas que os especialistas tinham para obter imagens, surgiu a Medicina Nuclear e com ela a modalidade PET como

uma inovadora modalidade de imagem médica, através da qual foi possível obter imagens funcionais do interior do organismo.

As primeiras tentativas não invasivas de obtenção de imagens datam de 1896, feitas pelo físico alemão Wilhelm Röntgen, usando uma radiação eletromagnética invisível (raios X). Röntgen percebeu que os raios X conseguiam penetrar o corpo e desta maneira era possível produzir uma representação do seu interior, fornecendo detalhes surpreendentes dos ossos, cavidades e outras estruturas anatômicas em seu caminho. No entanto, os raios X limitam-se a mostrar apenas estruturas anatômicas, e nada mais [Sabbatini, 1997].

Estudos pioneiros liderados pelo pesquisador Michael E. Phelps da Universidade da Califórnia foram os primeiros em evidenciar com excelente detalhe como diferentes partes do cérebro são ativadas quando desempenhamos tarefas mentais, tais como ouvir, ler, falar, pensar, etc. Estas tentativas iniciais foram efetuadas utilizando tomógrafos PET com evidentes limitações, por exemplo, com um número restrito de sensores para a reconstrução das imagens, os quais só conseguiam gerar apenas uma fatia de cada vez, com baixa qualidade e pobre definição [Robilotta, 2006].

Em 1975 aparecem os primeiros modelos comerciais de PET, sendo principalmente destinados a centros acadêmicos e de pesquisa [Caetano, 2002]. Posteriormente, a partir da segunda metade de 1980, aplicações clínicas utilizando esta modalidade foram se tornando mais evidentes e sua utilização foi mais intensiva nos diagnósticos médicos e nos estudos dinâmicos do metabolismo humano. Esse incremento na utilização da modalidade PET foi uma consequência, tal como acontece na atualidade, da maior facilidade de obtenção dos traçadores (produzidos próximo aos aparelhos PET) ou fornecidos por provedores regionais, evolução do hardware (anéis de sensores, etc.), programas de computação mais poderosos, entre outros.

A figura 1 mostra a evolução histórica das imagens PET desde sua origem:

FIGURA 1 – Evolução histórica da qualidade da imagem PET desde sua origem (Imagem Siemens, <http://www.cerebromente.org.br/>).

Os tomógrafos PET atuais (Figura 2) são bastante sofisticados, complexos e caros, mas ao mesmo tempo são muito mais rápidos, seguros e fáceis de operar, oferecendo um conjunto de características vantajosas usadas pelos médicos no processo de obtenção das imagens.

FIGURA 2 – O mais moderno aparelho da GE: Optima PET/CT 560 (Imagem GE Healthcare, <http://www.gehealthcare.com/>).

2.1.2 Processos Físicos na técnica PET

A Medicina Nuclear é uma especialidade médica que envolve o uso de materiais radioativos denominados radiofármacos no estudo, diagnóstico, terapia e no tratamento de várias condições clínicas e doenças [De Almeida, 2002].

Existem dois tipos de exames tomográficos utilizando o princípio de Medicina Nuclear, diferenciando-se uma da outra pela ocorrência de liberação de radiação gama: Quando existe emissão de pósitrons os quais interagem com os elétrons e geram 2 fótons simultâneos e em direções praticamente opostas, a técnica é chamada de **PET** (*Positron Emission Tomography*). Por outro lado, quando a emissão não é pósitron, mas sim fóton, como resultado do decaimento do radiofármaco, a técnica é chamada de **SPECT** (*Single Photon Emission Computed Tomography*).

A imagem obtida com o PET inicia-se com a administração ao paciente (por via endovenosa) de uma substância metabolicamente ativa, que carrega um isótopo emissor de pósitron (por exemplo, C11, N13, O15 ou F18), funcionando como um traçador [Oliveira et al., 2006]. Depois de alguns minutos, o isótopo vai se alojar na área do corpo onde a molécula tem afinidade. O núcleo radioativo instável decai a um estado mais estável pela emissão de elétrons carregados positivamente (isto é, pósitron). O pósitron ejetado combina-se com um elétron e quase instantaneamente essas duas partículas sofrem um processo de aniquilação por terem cargas iguais e opostas, emitindo sua energia de repouso sob a forma de dois fótons de alta energia (fótons 511 keV). Essa radiação gama emerge do corpo em direções praticamente opostas e são posteriormente detectados por uma combinação de cristais de cintilação que converte os fótons de aniquilação em luz visível, sendo que eventualmente ocorre um desvio dos fótons emitidos (fenômeno conhecido como dispersão ou scattering) que influirá na qualidade da

imagem. Na sequência, através de um tubo fotomultiplicador, é convertida a luz visível num sinal eletrônico originando finalmente a imagem médica [Wagner Jr., 1998].

Depois da detecção de um grande número de eventos produzidos pela aniquilação de pósitrons e elétrons, a distribuição do traçador é computacionalmente calculada a partir dos dados de projeção registrados. Por meio de um computador é processada essa distribuição, e através de procedimentos de reconstrução são obtidas as respectivas imagens transversas, coronais e sagitais [National Academy of Sciences, 1996; Wagner Jr., 1998].

Projeções unidimensionais obtidas em diferentes ângulos tornam possível reconstruir uma imagem bidimensional, assim como também se pode fazer reconstruções tridimensionais a partir de projeções bidimensionais [Fabey, 1996; Peregrino et al., 2004]. As imagens resultantes correspondem a medidas da distribuição do isótopo, podendo ser usadas tanto para avaliar, qualitativamente, o local de acúmulos não usuais do traçador (por exemplo, em tumores), quanto para medir, quantitativamente, a captação do traçador, para um diagnóstico mais profundo ou a localização de uma doença [Adams et al., 1999].

Na sequência vamos mencionar algumas das principais características que fazem parte desta modalidade, com a intenção de entender melhor todas as implicações no processo de geração das imagens PET:

Alcance do pósitron - Constitui um fator limitante na resolução da imagem obtida através desta modalidade. Os pósitrons saem do núcleo com certa energia que vai se perdendo com as contínuas colisões que experimentam com o meio que lhes rodeia. No momento que o pósitron perdeu total ou quase totalmente sua energia vai se aniquilar com um elétron, produzindo dois fótons colineares e de igual energia. A distância percorrida entre o ponto de emissão do pósitron e o ponto de aniquilação

é denominada como alcance do pósitron, tal como é apresentado no esquema da Figura 3 [Pozzo, 2005].

FIGURA 3 – Esquema da emissão e aniquilação do pósitron. A modalidade PET é baseada neste processo físico (Imagem adaptada de <http://www.hardmob.com.br/threads/438088-Aniquila%C3%A7%C3%A3o-de-mat%C3%A9ria-e-antimatéria-poder%C3%A1-criar-laser-de-Raios-Gama/>).

Não colinearidade - Uma vez ocorrida a aniquilação como se mostra na Figura 4, se o pósitron não tiver perdido toda sua energia cinética, o momento linear e a energia vão ser conservados, com perda da colinearidade dos dois fótons emitidos [López, 2008].

FIGURA 4 – Esquema de aniquilação com representação do efeito da não colinearidade [López, 2008].

Dispersão (scattering) e atenuação - Existe a probabilidade de que a radiação gama emitida como resultado da aniquilação do pósitron sofram alguma dispersão Compton (interferência com algum órgão ou estrutura intermediária). Neste caso, o fóton experimentará um desvio com um determinado ângulo α , portanto, incidirá num detector diferente ao detector que teria atingido sem dispersão [López, 2008]. Estes efeitos pioram a qualidade da imagem, devido à introdução de coincidências em linhas de resposta errôneas (Figura 5). Além do efeito Compton, se o fóton emitido sofre uma redução da sua intensidade original de radiação (fóton menos penetrante), produto da interação com os elementos do meio onde foi gerado, produz-se o que é conhecido como atenuação da imagem.

FIGURA 5 – Dispersão Compton no Objeto emissor [López, 2008].

Eletrônica - É vital ter em consideração a relação direta que tem a eletrônica com a obtenção de imagens PET de qualidade. Neste caminho, vêm acontecendo grandes esforços na tentativa de melhorar a resposta temporal dos detectores, o qual significaria um incremento da resolução na imagem. Deste modo, a concepção da reconstrução convencional mudou substancialmente com o surgimento de uma técnica conhecida como "Time-of-Flight", o qual proporciona informações para uma melhor localização do evento de aniquilação ao

longo da linha formada por cada par de detectores, capturando a diferença de tempo absoluta (em picosegundos) entre a detecção dos eventos coincidentes e a origem da aniquilação, tendo como resultado uma melhoria na relação sinal-ruído (SNR) da imagem reconstruída, traduzindo-se em melhoras consideráveis na qualidade das imagens [Matej et al., 2008].

Um tomógrafo PET pode ser descrito como uma estrutura móvel que se desliza ao longo do paciente, através de uma abertura circular (Figura 6). Ao redor da abertura, existem detetores de fótons, cujos sinais são amplificados por fotomultiplicadores e a informação é mandada ao computador que controla o aparelho. O computador trabalha os dados registrados e reconstrói o lugar estimado onde cada pulso de radiação se originou, mostrando a imagem como se fosse um corte (secção) através da área sob exame [Sabbatini, 1997]. Um conjunto destes cortes numa disposição adequada consegue construir a imagem tridimensional do órgão ou estrutura que está sendo analisada.

FIGURA 6 – Os cristais convertem a radiação gama, emitidos do paciente, a fótons de luz, e os tubos fotomultiplicadores convertem e amplificam os fótons a sinais elétricos. Estes sinais elétricos então, são processados pelo computador para finalmente gerar a imagem dessa região. O aparelho, então, pode ser movido, e com o processo repetido, obtém-se uma série de imagens (fatias) do corpo sobre a região de interesse (Imagem adaptada de <http://health.howstuffworks.com/medicine/modern/nuclear-medicine1.htm>).

Radioquímica - Os elementos envolvidos na obtenção das imagens baseadas na modalidade PET podem se dividir em:

(a) O Cíclotron ou acelerador destinado à produção dos radiofármacos necessários na utilização do PET no ambiente médico. Alguns radiofármacos (como o *rubídeo*⁸² e o *gálio*⁶⁸) podem ser obtidos através de geradores. Como equipamento, os geradores são mais simples e baratos do que os cíclotrons, em termos de nível de investimento e manutenção, sendo que este tipo de equipamentos pressupõe condições adequadas de operação (recinto blindado, procedimentos seguros, etc.). Outra característica importante que diferencia estes dois métodos, é o fato de que os cíclotrons têm a capacidade de produzir vários tipos de radiofármacos ao mesmo tempo, enquanto os geradores produzem um único radiofármaco, sendo que os isótopos produzidos dessa maneira são menos interessantes para estudos fisiológicos que aqueles produzidos por cíclotrons [Knapp e Mirazeh, 1994].

(b) Os Radiofármacos são moléculas que apresentam na sua constituição um isótopo radioativo que dependendo da sua distribuição, é usada para quantificar processos biológicos, permitindo realizar estudos metabólicos de uma zona específica do corpo, viabilizando o diagnóstico e/ou tratamento de doenças. Os radioisótopos mais utilizados na modalidade PET, produzidos por cíclotrons, são o Flúor¹⁸, o Carbono¹¹, o Oxigênio¹⁵ e o Nitrogênio¹³, com tempos de meia-vida de 110, 20, 2 e 10 minutos, respectivamente. Além desses, o rubídio⁸² é outro agente amplamente usado nos estudos do fluxo miocárdico, sendo produzido através de um gerador [De Amorin, 2007].

2.1.3 Comparação do PET com respeito às outras modalidades de imagens médicas

O princípio do PET é baseado na detecção coincidente dos fótons, produto da aniquilação pósitron – elétron, emitidos em

direções opostas. Ao realizar uma medição da radiação gama com suficiente amostragem espacial, e mediante o emprego de técnicas de reconstrução de imagens, pode-se obter a distribuição espacial do marcador dentro do organismo em estudo. Adicionalmente, se o estudo é realizado em vários intervalos sucessivos de tempo, é obtida uma distribuição temporal de imagens, possibilitando estudos dinâmicos.

Na Tabela 1 são apresentados alguns dos radiofármacos usualmente utilizados nos exames PET, assim como sua função biológica [Pozzo, 2005; Robilotta, 2006].

RADIOFÁRMACOS	INDICAÇÕES
Marcados com 18F	
2-[18F]fluoro-2desoxi-D-glicose (18FDG)	Metabolismo da glicose
18-F-Fluoroestradiol	Densidade de receptores hormonais no câncer de mama
18-Fluoruro	Metabolismo Ósseo
18F-Fluorouracilo	Comportamento do quimioterápico não marcado
18F-L-DOPA	Função dopaminérgica presináptica
18F-Tamoxifeno	Comportamento do quimioterápico não marcado
18F-Fluorodesoxiuridina	Sínteses do ADN
Marcados com 11C	
11C-Metionina	Transportadores de aminoácidos e síntese de proteínas
11C-Tirosina	Transportadores de aminoácidos e sínteses de proteínas
11C-Leucina	Transportadores de aminoácidos e sínteses de proteínas
11C-Timidina	Sínteses do DNA
11C-Acetato	Metabolismo oxidativo miocárdico
11C-Flumazenil	Receptores de benzodiazepinas
11C-Raclopride	Receptores D2
11C-Hidroxi-Efedrina	Reinervação de trasplante cardíaco
11C-N-Metil-4-Piperidil Acetato	Atividade de acetilcolinesterase cerebral
11C-Tezolomida	Comportamento do quimioterápico não marcado
11C-PK 11195	Marcador da atividade da microglia
Marcados com 15O	
15O-Água	Fluxo sanguíneo regional tumoral e a neovascularização associada a determinados tumores como os cerebrais
15O-Butanol	Fluxo sanguíneo cerebral
Marcados com 13N	
13N-Glutamato	Transportadores de aminoácidos e sínteses de proteínas
13N-Cisplatino	comportamento do quimioterápico não marcado
13N-Amonio	Fluxo sanguíneo miocárdico

TABELA 1 – Listado dos radiofármacos comumente empregados na modalidade PET [Pozzo, 2005].

O PET é uma modalidade de imagem capaz de medir processos bioquímicos dentro dos tecidos, permitindo a investigação da perfusão, dos caminhos metabólicos e dos processos biomoleculares em

estados normais e patológicos, fornecendo imagens que demonstram alterações bioquímicas mesmo onde não existe uma anormalidade estrutural evidente [Jones et al., 1996; Blue Cross e Blue Shield, 2001].

Pode-se então afirmar, que a principal diferença entre a Tomografia Computadorizada (CT) e o PET não é tecnológica. Essa diferença reside na capacidade de detectar e caracterizar a doença por suas características biológicas em oposição à anatomia. Diferentemente da CT que é uma técnica de transmissão, o PET é uma técnica de mapeamento das emissões dos elementos radioativos de dentro para fora [Jones et al., 1996; Rabilotta, 2006].

Na Tabela 2 é mostrada uma comparação da máxima resolução alcançável, não necessariamente em aplicações clínicas rotineiras, com cada modalidade, assim como as limitações de profundidade e tempo que as mesmas apresentam [López, 2008].

MODALIDADE	RESOLUÇÃO	PROFUNDIDADE	TEMPO
Ressonância Magnética (RMI)	10 - 100 μ m	-	Min
Tomografia Computadorizada (CT)	50 μ m	-	Seg
Ultra - Sonografia (US)	1 mm	cm	Seg
Tomografia por Emissão de Pósitrons (PET)	1 - 2 mm	-	Min
Tomografia Computadorizada por Emissão de Fóton único (SPECT)	< 1 mm	-	Min
Tomografia por Emissão de Fótons (FMT)	1 - 2 mm	< 10 cm	Seg

TABELA 2 – Comparação das principais características entre as modalidades de Imagens Médicas existentes [López, 2008].

2.1.4 Principais aplicações do PET

As imagens obtidas através da modalidade PET vêm sendo utilizadas para avaliar uma variedade de doenças, com base em seus processos funcionais, metabólicos, bioquímicos e fisiológicos. Basicamente suas principais aplicações clínicas ficam concentradas em três grandes áreas: oncologia, cardiologia e desordens neuropsiquiátricas.

(i) PET em Oncologia

O uso do PET em oncologia tem crescido significativamente nos últimos anos, principalmente desde o desenvolvimento dos equipamentos que fizeram possíveis estudos do corpo inteiro. A grande vantagem das imagens obtidas através do PET é a possibilidade de detectar a atividade de massas muito pequenas de células cancerosas, que além disso, refletem fielmente a atividade tumoral [Adams et al., 1999].

É conhecido que a CT e a MRI são técnicas de imagem basicamente anatômicas de alta resolução muito utilizadas em oncologia para detectar e confirmar a presença de elementos tumorais, assim como fornecer informações sobre o tamanho, a localização e seu grau de disseminação e/ou extensão do tumor [Oncoguia, 2009]. No entanto, métodos como o PET estabelecem parâmetros metabólicos ou funcionais que podem auxiliar na distinção de áreas cancerígenas que habitualmente tem maior concentração dos traçadores por terem um metabolismo mais intenso comparado com o tecido circunvizinho, mesmo numa fase inicial. Esta grande vantagem faz que o PET seja uma técnica mais acurada para o diagnóstico, estadiamento e decisões de tratamento em oncologia [López, 2008].

A utilidade dos estudos PET se tem demonstrado em diferentes tipos de tumores, entre os quais se podem citar o câncer de mama, câncer colo retal, tumores cerebrais, linfomas, câncer de pulmão, câncer de cabeça e pescoço, câncer de esôfago, câncer de tireóide, câncer de ovário e câncer de pâncreas [Robilotta, 2006].

(ii) PET em Cardiologia

Em cardiologia, as aplicações potenciais do PET ficam principalmente concentradas na avaliação da viabilidade miocárdica, em pacientes com doença e com disfunção cardíaca. Os estudos PET permitem distinguir o miocárdio isquêmico viável do necrótico. Se for

observada uma diminuição do metabolismo, pode-se confirmar a existência de tecido miocárdio não viável, que não se beneficiará do restabelecimento do fluxo sanguíneo. Caso contrário, a demonstração no tecido de um metabolismo normal indica sua viabilidade e assegura a melhoria que será obtida com o restabelecimento do fluxo sanguíneo. A grande importância que isso representa, é que permite uma seleção eficiente dos pacientes da lista de espera de transplantes para processos de revascularização, evitando angiografias, angioplastias e pontes de safena [Robert e Milne, 1999].

(iii) PET em Neurologia

No início, a maioria das aplicações clínicas e de pesquisa utilizando o PET concentrava-se na análise de distúrbios neurológicos e psiquiátricos, fazendo possíveis estudos de lesões não detectáveis com os procedimentos anatômicos [Robilotta, 2006]. Este tipo de exames fornece informações valiosas sobre propriedades bioquímicas e fisiológicas dos tecidos normais e doentes, determinando as regiões com lesões cerebrais.

As aplicações clínicas potenciais do PET em distúrbios neuropsiquiátricos, principalmente se concentram em [Wagner Jr., 1998]:

- Avaliação de tumores cerebrais primários.
- Diagnóstico de demência (doença de Alzheimer).
- Avaliação pré-cirúrgica de epilepsia, com localização não invasiva de focos epiletogênicos em pacientes com epilepsia parcial.
- Biópsia guiada em tumores cerebrais.
- Seleção de pacientes com autismo ou transtornos generalizados do crescimento (falha maturativa nos circuitos funcionais do tálamo).

2.3 Presente e futuro do PET

2.3.1. Situação atual no Brasil

Os atuais aparelhos de Medicina Nuclear disponíveis nos centros médicos, em particular aqueles do tipo dedicado para exame de corpo inteiro, possuem uma satisfatória resolução e qualidade de imagem para as principais indicações clínicas, mas avanços quanto a esses parâmetros continuam sendo uma demanda técnica importante, o que tem estimulado os fabricantes dos equipamentos PET a focar sua pesquisa e desenvolvimento em instrumentação e novos softwares, que ajudem a melhorar o desempenho, reduzam custos e tornem a operação dos sistemas mais “amigáveis” aos usuários [Budinger, 1998].

Mas, qual é a verdadeira razão pela qual não existem tantos scanners PET comparado com outros aparelhos de imagens espalhados pelo mundo e pelo Brasil, tendo em consideração que um equipamento PET não é mais complexo do que seus pares de outras modalidades? A resposta se baseia nos radiofármacos (principal desvantagem desta modalidade), os quais devido à meia-vida curta dos emissores de pósitrons, requerem que sua produção deva ser realizada próxima ao local de aplicação. Nesse sentido, a instalação de um ciclotron e seus acessórios exigem um investimento considerável além de profissionais especializados, o que pressupõe dispor de um orçamento significativo.

Mesmo assim, há iniciativas importantes, como do centro produtor de radioisótopos do Instituto de Engenharia Nuclear (IEN) no Rio de Janeiro, que iniciou a produção diária e regular de FDG em 2002. Iniciativas semelhantes se desenvolvem em São Paulo, que já conta com a produção de FDG no Instituto de Pesquisas Energéticas e Nucleares (IPEN) desde 1999 [Araújo et al., 2008] e vários sistemas PET-CT instalados.

Projetos análogos estão sendo estudados em outros pólos médicos como Recife, Porto Alegre, Belo Horizonte, entre outras capitais de estado em todo o Brasil [Robilotta, 2006].

Embora se reconheça o imenso potencial clínico da modalidade PET, os altos custos de aquisição e operação exigem uma definição clara da sua utilidade em todos os campos da medicina, estabelecendo os limites do método, cotejada com as outras modalidades e/ou tecnologias concorrentes, com custos mais acessíveis. Na Figura 7, pode-se apreciar um comparativo do volume dos procedimentos das principais modalidades de diagnóstico por imagem no percurso do tempo. Claramente, os pontos debatidos anteriormente repercutem significativamente no número de procedimentos, utilizando a modalidade PET em relação aos outros procedimentos de obtenção de imagens médicas.

FIGURA 7 – Comparativo do volume dos procedimentos de diagnóstico de imagem por modalidade (Imagem adaptada de http://maxbley.typepad.com/maxs_blog/2007/09/if-im-perfectly.html).

Pode-se indicar que os estudos e análises realizados com a modalidade PET não são uma utopia, são:

1. Alternativas verdadeiramente viáveis e eficientes no diagnóstico e tratamento do câncer e outras doenças.

2. Ferramentas com um grande valor no gerenciamento das doenças, fundamentais nas decisões precoces e corretas por parte dos especialistas. Importante na redução de riscos cirúrgicos, tratamentos intensivos e longas convalescenças.
3. Instrumento promissor da medicina preventiva.

2.3.2. Perspectivas no futuro

O futuro do PET é promissor, entre outras coisas se estão desenvolvendo novos materiais detectores, com maior eficiência e precisão. Além disso, as técnicas de reconstrução estão evoluindo, com o objetivo de criar máquinas com velocidade e capacidade superiores, a fim de permitir reconstruções volumétricas em tempo real. No entanto, a parte mais importante para o progresso futuro do PET é o desenvolvimento de novos radiofármacos. Quando dispusermos de agentes tumorais mais específicos, se estenderá um campo de aplicação com agentes diagnósticos específicos para cada tipo de tumor.

Em particular o uso de sistemas combinados como o caso do PET/CT está propiciando uma interação interessante entre médicos nucleares e radiologistas, permitindo ter um conhecimento mais claro das estruturas anatômicas e melhor compreensão das suas inter-relações funcionais [Robilotta, 2006].

Por outro lado, a necessidade de maior definição anatômica tem levado à introdução de procedimentos que vinculam modalidades de obtenção de imagens (sistemas híbridos), permitindo a fusão de dados em imagens anatômicas (CT) e fisiológicas (PET), o qual está sendo muito favorável nas diversas aplicações médicas atuais, permitindo fazer diagnósticos mais precisos e planejar os tratamentos mais adequados com menos efeitos secundários [Bruce, 2000].

Ao redor do mundo, os obstáculos para o crescimento e consolidação do PET estão sendo ultrapassados. O número de centros

que contam com equipamentos PET cresce exponencialmente, não só nos Estados Unidos, Europa e Japão, mas em países com economia menos privilegiada, como Arábia Saudita, Turquia, Coréia do Sul, República Checa e Taiwan [Peregrino et al., 2004]. Na América Latina, além do Brasil, há scanners PET instalados no México, na Argentina e no Chile. Como dado adicional, pode-se indicar que no Peru o Instituto do Coração, que foi inaugurado em março de 2010 na cidade de Lima, ainda não tem nenhum equipamento de Medicina Nuclear, mas se prevê em breve a chegada e o uso desta modalidade pelas importantes vantagens que traz.

2.4 Segmentação de Imagens Digitais

O processo de segmentar um objeto dentro de uma imagem é uma tarefa complicada para um computador, quando a característica que difere entre o objeto e o background é alguma propriedade da textura em vez da intensidade, ou quando a imagem está corrompida por ruído ou por uma iluminação não homogênea [Carvalho et al., 1999]. A existência dessas interferências na imagem exige dos algoritmos de segmentação um esforço adicional para identificar e separar os objetos ou regiões de interesse presentes nas imagens [Gonzalez; Woods, 2008]. Na literatura encontramos uma variedade de métodos para este fim, entre os mais conhecidos pode-se mencionar os seguintes: Snakes [Kass et al., 1988; Xu, 1998], Level Set [Osher; Fedkiw, 2002], Fuzzy Connectedness [Nyúl et al., 2002; Udupa; Saha, 2003; Pednekar; Kakadiaris, 2006], IFT [Falcão et al., 2004], Region Growing [Gonzalez; Woods, 2008], thresholding [Otsu, 1979; Gonzalez; Woods, 2008], entre outras.

As investigações neste trabalho têm como objetivo analisar e aplicar algoritmos de segmentação baseados no conceito de Fuzzy Connectedness (FC). O método Fuzzy Connectedness tem se mostrado

eficiente na segmentação de imagens médicas, mesmo comparado com outros métodos [Kass et al.; 1988; Xu, 1998], e dependendo da aplicação, ele é mais interessante, pois é um método robusto, principalmente quando se trata de imagens com intensidades não-uniformes [Pednekar; Kakadiaris, 2006].

A Figura 8 mostra um exemplo de segmentação utilizando os métodos Region Growing e Fuzzy Connectedness [Udupa; Saha, 2003]. Em (a) se tem a imagem original simulada, em (b) na mesma imagem foi incorporado ruído gaussiano, em (c) é mostrado o mapa de afinidades após a execução do FC. Na sequência, foi aplicado um thresholding, tendo como resultado a imagem (e), já em (d) foi aplicado o Region Growing, que apresentou ter perda de informação, comparado com o FC, como pode ser observado.

Figura 8 – Exemplo de segmentação do círculo na imagem simulada para demonstrar a efetividade da Conectividade Fuzzy em relação a um método mais simples, como por exemplo o Region Growing - (a) Imagem simulada original, (b) Imagem com ruído Gaussiano, (c) Relação de afinidade de cada spel, (d) Círculo segmentado por crescimento de região (percebe-se que uma parte dele foi perdida), (e) Círculo segmentado por Conectividade Fuzzy [Udupa; Saha, 2003].

Uma etapa inicial, como parte da investigação, foi o levantamento bibliográfico relacionado à teoria FC e às aplicações dessa técnica no auxílio da resolução do problema de segmentação das imagens médicas PET.

2.4.1. Definições iniciais da Teoria Fuzzy

Etimologicamente a palavra “fuzzy” é associado a termos como nebuloso, difuso, vago ou incerto [Etymology Dictionary, 2011]. O principal conceito da Teoria Fuzzy é o fato de dar flexibilidade na modelagem, permitindo desenvolver algoritmos semelhantes ao pensamento humano [Souza; De Oliveira, 1992].

A Teoria Fuzzy utiliza, como princípio fundamental de execução, uma função de associação ou de vínculo que envolve elementos específicos dentro de uma imagem [Furuie et al., 1996]. Baseados nestes vínculos, pode-se estabelecer uma classificação de relação entre os elementos que pertencem, ou não, a um objeto específico [Nyúl et al., 2002; Udupa; Samarasekera, 1996; Udupa; Saha, 2003].

2.4.2. Espaço Digital Fuzzy

Tendo o conjunto Z como a representação dos números inteiros, então o conjunto Z^n seria uma representação de um espaço digital n -dimensional, cujos elementos são denominados spels (Spatial Elements). No caso de uma imagem bidimensional (onde $n=2$) passam a ser chamados pixels, já para imagens tri-dimensionais (onde $n=3$), como é nosso caso, são chamados voxels [Gonzalez; Woods, 2008; Nyúl et al., 2002].

Uma relação fuzzy ρ é um subconjunto fuzzy de Z^n , definido por $\rho = \{((x,y), \mu_\rho(x,y)) | x,y \in Z^n\}$, onde $\mu_\rho(x,y): Z^n \times Z^n \rightarrow [0,1]$ representa uma função de pertinência, cujo valor indica o grau de pertinência dos elementos (x,y) dentro do subconjunto ρ [Shaw; Simões, 1999; Gonzalez; Woods, 2008].

A relação ρ é reflexiva, se, $\forall x \in Z^n$, $\mu_\rho(x, x) = 1$; simétrica, se, $\forall x, y \in Z^n$, $\mu_\rho(x, y) = \mu_\rho(y, x)$; e transitiva se $\forall x, z \in Z^n$, $\mu_\rho(x, z) = \max_{y \in Z^n}(\min[\mu_\rho(x, y), \mu_\rho(y, z)])$. Caso as três propriedades tenham ocorrência simultânea, ρ é chamada de relação de semelhança. A vizinhança fuzzy cumpre as propriedades reflexiva e simétrica simultaneamente [Lage, 2010; Udupa; Samarasekera, 1996; Udupa; Saha, 2003; Gonzalez; Woods, 2008].

Considerando as definições anteriores, o par (Z^n, α) , onde α é uma relação de vizinhança fuzzy é chamado Espaço Digital Fuzzy.

2.4.3. Cena de Pertinência

Uma cena definida em (Z^n, α) é um par $C = (\mathcal{C}, f)$, onde $\mathcal{C} = \{c | -b_i \leq c_i \leq b_i, b \in Z_+^n\}$ é o domínio da cena, f , chamada intensidade da cena, é uma função definida como $f: \mathcal{C} \rightarrow [L, H]$, onde $[L, H]$ é um conjunto de números, geralmente inteiros que representam os valores mínimo e máximo, dentro dos quais estão contidas as intensidades de cada um dos elementos [Nyúl et al., 2002]. Com essas premissas, denominamos cena de pertinência ao subconjunto fuzzy definido no domínio da imagem obtido como resultado de uma segmentação, cuja representação equivalente em cena é tal que a intensidade da cena age como uma função de pertinência fuzzy, com valores compreendidos entre $[0, 1]$ [Lage, 2010].

2.4.4. Adjacência e Afinidade Fuzzy

A relação de adjacência fuzzy entre cada par de spels, (c, d) , definidos no domínio do espaço digital, indica a proximidade espacial entre os dois elementos comparados, atribuindo valores entre zero e um [Nyúl et al., 2002]. Se os elementos c e d fossem vizinhos seria

uma condição suficiente para que tivessem adjacência Fuzzy unitária. Genericamente, a relação de adjacência é representada por α e o grau de pertinência por $\mu_\alpha(c, d)$ [Lage, 2010; Udupa; Saha, 2003; Udupa; Samarasekera, 1996; Nyúl et al., 2002].

Agora, uma relação de afinidade fuzzy, definida no domínio de uma cena C , para cada par de spels (c, d) , representa a similaridade local existente entre as intensidades dos spels comparados. Genericamente, a relação de afinidade é representada por k e o grau de pertinência dessa relação é definido por $\mu_k(c, d)$ [Lage, 2010; Udupa; Saha, 2003; Udupa; Samarasekera, 1996; Nyúl et al., 2002]. O conceito de afinidade num processo de segmentação é fundamental na obtenção de resultados confiáveis e eficientes [Lage, 2010].

2.4.5. Conectividade Fuzzy e Objeto Fuzzy

Partindo dos conceitos de adjacência e afinidade fuzzy abordados, é possível encontrar uma conectividade entre quaisquer dois spels, c e d , no domínio da cena C , a partir de uma série de afinidades locais entre o par de elementos [Nyúl et al., 2002]. Fica claro que existem vários caminhos passando somente por adjacências unitárias que consigam unir esses dois spels. A uma conectividade global no domínio C dá-se o nome de Fuzzy Connectedness, representada por K e sua função de pertinência entre c e d por $\mu_K(c, d)$.

Assumindo que $p_{c,d} = \langle s_1, s_2, \dots, s_N \rangle$ seja uma caminho entre $s_1 = c$ e $s_N = d$, onde s_i são N spels adjacentes em pares entre si, a afinidade desse caminho é dada por:

$$\mu_K(p_{c,d}) = \min_{1 \leq i \leq N} (\mu_k(s_{i-1}, s_i)) \quad (2.1)$$

Então, a equação (2.1) mostra cada caminho entre dois elementos (c, d) a qual estará definida, representada ou limitada pela afinidade mais fraca do conjunto de afinidades que esse caminho possua. Como foi dito, são inúmeros os caminhos entre dois spels e como é evidente num caso ótimo, a conectividade global deverá considerá-los todos. Com tudo isso, a seleção da maior afinidade de caminho será a melhor representação da conectividade global. Portanto, assumindo que $P_{c,d}$ seja um conjunto de todos os caminhos p , a conectividade global entre c e d é definida como:

$$\mu_K(c, d) = \max_{p \in P_{c,d}} (\min_{1 \leq i \leq N} (\mu_K(s_{i-1}, s_i))) \quad (2.2)$$

Dessa forma, a quantificação da função de pertinência de uma conectividade é totalmente dependente da definição de afinidade local entre os elementos que a compõem [Lage, 2010].

Capítulo 3

METODOLOGIA

Nesse capítulo são apresentados com detalhes apropriados cada um dos blocos dentro da sequência de processamento utilizada, assim como as escolhas feitas como parte de nossa pesquisa.

3.1 Conceitos Iniciais

Processar uma imagem consiste na sua manipulação através de técnicas que objetivam a extração da informação nela contida [Gonzalez; Woods, 2008], visando uma melhoria de informação visual para interpretação humana ou para alguma aplicação específica.

Entre o conjunto de tarefas que conformam o processamento das imagens, existe uma inter-relação estreita entre cada um dos passos envolvidos neste processo, já que cada uma destas etapas contribui a que as imagens que estão sendo analisadas se tornem adequadas, ou pelo menos com melhores características para o sucesso do processo ou etapa posterior.

Isso é bastante lógico, porque como é conhecido, as imagens de Medicina Nuclear possuem características particulares no momento da sua aquisição, as quais deverão ser tratadas de alguma maneira com o objetivo de facilitar e fazer muito mais eficientes os resultados obtidos ao final da sequência de processamento.

3.2. Diagrama de Blocos

O diagrama de blocos a seguir resume de forma geral as partes da sequência de processamento que faz parte deste trabalho.

FIGURA 9 – Diagrama de blocos geral do projeto.

Entre os principais blocos que conformam nosso projeto e que foram apresentados na figura anterior podemos mencionar os seguintes:

- Aquisição e/ou Simulação de Imagens tridimensionais de Medicina Nuclear;
- Adição de Ruído quântico tipo Poisson nas imagens simuladas;
- Pre-processamento: Filtragem de imagens tridimensionais;
- Segmentação de estruturas de interesse nas imagens tridimensionais;
- Pós filtragem através de morfologia matemática;
- Quantificação dos volumes;
- Análise dinâmica de estruturas 3D;
- Avaliação da quantificação e das análises dinâmicas.

O diagrama da Figura 9 ilustra o processo e os módulos a serem desenvolvidos para a quantificação e a análise da dinâmica a partir das imagens 3D reconstruídas de PET. Para a análise temporal, uma sequência (uniforme ou não) de imagens funcionais 3D terá suas estruturas identificadas, segmentadas e quantificadas.

3.3. Aquisição/Simulação das imagens 3D

A aquisição de uma imagem está ligada diretamente ao processo de captura da imagem, usando um método específico [Gonzalez; Woods, 2008]. É importante, na maneira do possível, que haja uma minimização dos reflexos na etapa de aquisição a fim de reduzir o ruído produzido na imagem. Obviamente é fundamental que as imagens capturadas sejam de alta qualidade [Russ, 1998], de tal forma que a informação contida nela seja fácil e claramente interpretada.

Na investigação proposta não se envolvem novos experimentos com animais ou seres humanos no processo de obtenção das imagens que serão analisadas. Nesse sentido, utilizaremos dados retrospectivos, isto é, sequência de imagens 3D arquivadas no prontuário eletrônico do Instituto do Coração (InCor) do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo (HC-FMUSP) e/ou *phantoms* numéricos.

Existe uma variedade de processos que podem ser utilizados nesta primeira etapa, no entanto, para nosso estudo serão válidos só dois conjuntos de bases de imagens:

- Utilizou-se uma imagem de PET dinâmico cardíaco, adquirida no InCor-HCFMUSP, como parte de projeto de pesquisa protocolado pela Comissão de Ética para Análise de Projetos de Pesquisa do HCFMUSP (Cappesq), com número 0630/07.
- Assim também, imagens simuladas da região requerida serão utilizadas para realizar os estudos propostos pelo projeto. Por um lado, existe a

possibilidade de usar ferramentas conhecidas para a simulação e obtenção de volumes fisiológicos (imagens 3D). Entre os softwares conhecidos no meio clínico, o InCor - HCFMUSP possui algumas simulações feitas através da ferramenta NCAT-4D (Nurbs-based Cardiac-Torso), as quais serão utilizadas nesse projeto. Além disso, imagens 3D próprias geradas através da utilização de algum entorno de programação, serviram como volumes de entrada na sequência de processamento que foi mostrada na Figura 9.

Estes *phantoms* cumprem um papel cada vez mais preponderante nas pesquisas com imagens médicas, já que têm a capacidade de simular um número ilimitado de anatomias conhecidas dos pacientes, oferecendo meios práticos para avaliar quantitativamente, comparar e melhorar técnicas e dispositivos de imagens médicas [Segars; Tsui, 2009]. A seguir, apresentamos algumas das mais importantes características e detalhes dos *phantoms* utilizados neste trabalho:

(a) NCAT (Nurbs-based Cardiac-Torso)

O NCAT-4D foi desenvolvido para a pesquisa de imagens de medicina nuclear, especificamente para tomografia computadorizada por emissão de fóton único (SPECT) e tomografia por emissão de pósitrons (PET) [Segars; Tsui, 2009].

O NCAT-4D surgiu como uma versão melhorada do seu antecessor o MCAT-4D, tendo o claro propósito de criar um melhor phantom para pesquisa de imagens médicas, baseado numa nova base que iria nos permitir utilizar os mesmos recursos do MCAT-4D, mas com resultados da modelagem muito mais realistas da anatomia humana [Segars; Tsui, 2009]. Esta ferramenta fornece um modelo realista da anatomia humana (Figura 10), envolvendo movimentos cardíacos e respiratórios, fazendo possível a simulação de uma série de *phantoms* de três dimensões (3D) durante um determinado período de tempo, representando com cada phantom gerado um momento determinado do corpo enquanto este se

movimenta. Tanto o período de tempo, o número de *phantoms* de saída, quanto o tipo de movimentação requerido são definidos pelo usuário através de parâmetros determinados. Além da movimentação, este tipo de ferramenta tem a capacidade de modelar as variações anatômicas tanto masculinas quanto femininas. Desta maneira, indivíduos femininos são modelados de forma específica, por exemplo, pela simples adição de extensões de mama na anatomia do tórax masculino. Com esta capacidade de mexer na anatomia, o NCAT-4D pode ser usado para simular uma população de pacientes envolvidos em algum tipo específico de estudos [Segars; Tsui, 2009].

Na atualidade é bastante usado na pesquisa para avaliar e melhorar técnicas de aquisição, métodos de reconstrução e diversas aplicações com imagens médicas, especialmente aplicações cardíacas dinâmicas. No entanto, uma limitação importante do NCAT 4D é que não tem a capacidade de simular realisticamente movimentos anormais, resultado de algum tipo de patologia, tal como a doença arterial coronariana (CAD do inglês coronary artery disease) [Veress et al., 2006].

Figura 10 - (A) Vista anterior do phantom NCAT (3D), mostrando a segmentação detalhada de órgãos. (B) Imagem 3D do modelo cardíaco do phantom NCAT, com a descendente anterior esquerda (do inglês left anterior descending - LAD) e circunflexa esquerda (do inglês left circumflex - LCX), ramos da artéria coronária sobreposta à superfície exterior do modelo do coração. (C) Imagem 3D do modelo do ventrículo esquerdo do phantom NCAT mostrando as superfícies internas e externas, a LAD e LCX, os ramos da árvore arterial coronária, uma placa aterosclerótica e um simulado de um infarto do miocárdio. Fonte das imagens do paper: Normal and Pathological NCAT Image and Phantom Data Based on Physiologically Realistic Left Ventricle Finite Element Models [Veress et al., 2006].

(b) Imagens 3D do Ventrículo Esquerdo

O foco de nosso estudo é o Ventrículo Esquerdo (VE) do Coração (Figura 11), não só porque tem despertado interesse em pesquisas médicas pelo fato de representar um grave problema de saúde pública, refletido estatisticamente num número considerável de pacientes com CAD e insuficiência cardíaca (com risco de mortalidade cinco a oito vezes maior do que a população normal [Pedone et al., 2004]). Além disso, se analisarmos internamente as partes do coração e cada uma das suas funções, o VE representa a cavidade que realiza o “trabalho pesado” no processo de irrigação do sangue oxigenado, desde o coração até cada uma das partes do corpo humano, envolvendo parâmetros como pressões, velocidades, forças, etc., com magnitudes maiores em relação do que as apresentadas pelas outras cavidades do coração.

Por esse motivo, técnicas de diagnóstico por imagem desempenham um papel vital na redução da mortalidade, criando métodos mais eficientes para detectar, controlar e tratar pacientes com doenças desse tipo [Veress et al., 2006]. A função do Ventrículo Esquerdo (VE) é um importante indicador para o diagnóstico e prognóstico dos pacientes com CAD [Johnson et al., 1991; Shaw et al., 1998].

FIGURA 11 – *Phantom* tri-dimensional do Ventrículo Esquerdo gerado como parte do projeto, usando código em MatLab e reconstruído através do ImageJ.

3.4. Filtragem de Imagens 3D

Podemos definir o ruído numa imagem digitalizada, como toda flutuação indesejável das intensidades dos pixels que pode comprometer tanto a sensibilidade quanto a qualidade da imagem [Shrout, 1993; Sarmiento et al., 2004].

A restauração de uma imagem através da remoção do ruído contido nela é essencial em aplicações com imagens médicas, e consiste basicamente em utilizar informações conhecidas a priori sobre as degradações da imagem, de modo que seja possível aplicar um processo inverso para remover essas distorções a fim de melhorar e recuperar os detalhes anatômicos que podem estar escondidos nos dados [Rodrigues et al., 2008].

Num processo de filtragem, o valor de intensidade de uma imagem depende não só da intensidade original, mas dos valores de intensidade de sua vizinhança. O tamanho desta vizinhança não é limitado, mas normalmente os vizinhos mais próximos têm maior influência que aqueles que estão mais distantes. De modo geral, os filtros são projetados e implementados como recursos ou ferramentas que permitem aumentar o brilho eo contraste, bem como para adicionar uma grande variedade de texturas, tons e efeitos especiais numa imagem. Esta atividade repercute no incremento da qualidade geral da imagem, o que pode ser interessante para fins de visualização, e ainda para ressaltar determinados objetos ou características da imagem [Gonzalez; Woods, 2008].

Segundo a literatura, as imagens PET são corrompidas pelo ruído quântico que é dependente do sinal e pode ser modelado por uma distribuição estatística de Poisson [Bethea; Duran, 1995; Romualdo, 2009]. Por se tratar de um ruído aleatório, a modelagem do ruído quântico é baseada apenas em ferramentas estatísticas.

A distribuição estatística de Poisson é definida como a distribuição de probabilidade discreta que expressa a probabilidade de certo número

de eventos ocorrerem num intervalo de tempo ou espaço específico [Leitão; Mariz, 2010]. A probabilidade de que existam exatamente k ocorrências (sendo k um número inteiro não negativo) é dada pela equação seguinte:

$$f(k, \lambda) = P(X = k) = \frac{e^{-\lambda t} (\lambda t)^k}{k!} \quad (3.1)$$

onde e é a base do logaritmo natural ($e = 2,71828 \dots$), $k!$ é o fatorial de k , λ é um número real igual à taxa média do processo e t é o intervalo de tempo ou espaço.

Conforme a distribuição estatística de Poisson há variação no número real de fótons absorvidos por unidade de área do receptor da imagem, como ilustrado na Figura 12.

FIGURA 12 – Representação do ruído Poisson: Em (a) se mostra a imagem original do VE de entrada, já em (b) temos a imagem com o ruído Poisson incorporado (Typical PET image - <http://courses.washington.edu/>).

Uma abordagem que busca determinar quantitativamente e ao mesmo tempo busca garantir que o ruído incorporado nos volumes simulados e/ou *phantoms* corresponde efetivamente ao ruído Poisson, típico das imagens de Medicina Nuclear, é proposto de maneira detalhada na Secção 4.2.1.

Uma das técnicas de processamento comumente utilizadas para melhorar a qualidade de uma imagem é a aplicação de algoritmos que permitam eliminar ruídos indesejáveis na imagem, geralmente gerados no processo de aquisição [Gonzalez; Woods, 2008]. O objetivo deste processo de filtragem é conseguir uma imagem resultante com características mais adequadas que a imagem original para uma aplicação específica [Marques Filho; Vieira, 1999].

Os métodos de filtragem abordados, analisados, implementados e discutidos neste projeto são: o Filtro da Mediana, o Filtro da Gaussiana Ponderada e o Filtro de Anscombe/Wiener.

(i) **Filtro da Mediana**

Este método de filtragem é bastante utilizado em aplicações onde é preciso filtrar o ruído e ao mesmo tempo preservar as bordas e os detalhes característicos da imagem [Romualdo, 2009]. Basicamente o que esta técnica faz é substituir a intensidade do pixel central da janela pela mediana dos pixels situados em sua vizinhança, a qual vem definida pela dimensão da máscara escolhida [Marques Filho; Vieira, 1999].

Dada uma coordenada (i, j) na matriz A de inteiros positivos com m linhas e n colunas, a vizinhança de tamanho $p \times q$ está definida como uma submatriz $A_{i,j}$, com p linhas e q colunas (sendo p e q inteiros positivos ímpares) com centro em (i, j) . Este filtro calcula uma matriz Med com o mesmo tamanho de A , de forma que $Med(i, j)$ contém a mediana dos números em $A_{i,j}$ (a vizinhança $p \times q$ em torno de (i, j)) [Pertusa, 2003].

(ii) **Filtro da Gaussiana Ponderada**

Esta abordagem tem um comportamento particular comparado com os outros métodos clássicos aplicados em imagens. A filtragem de imagens usando o conceito da Gaussiana Ponderada se comporta como uma função matemática (expresada pela equação 3.2) aplicada aos

dados do perfil. O filtro gaussiano é uma média ponderada do perfil, onde os valores dos pesos se distribuem na forma gaussiana (distribuição normal). Esse filtro tem uma função interpoladora, ou seja, ele ameniza a amplitude do sinal dando mais suavidade aos contornos da superfície [Whitehouse, 1998; Pertusa, 2003].

A função de ponderação para o filtro gaussiano possui a equação de probabilidade gaussiana.

$$f(x) = \left[\frac{1}{\sigma\sqrt{2\pi}} \right] \cdot e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \quad (3.2)$$

onde σ é o desvio padrão da distribuição com valor sigma considerado no processo da filtragem. Espera-se que o valor de sigma esteja na faixa entre 0 e 3, que constituem os valores representativos dentro da curva de distribuição normal.

(iii) Filtro de Anscombe / Wiener (Lee)

O principal método de filtragem proposto neste trabalho efetua uma restauração nas imagens utilizando, numa primeira etapa, a transformada de Anscombe (AT) e o filtro de Wiener pontual (Lee) para redução do ruído quântico. Posteriormente, é utilizada a Transformada Inversa de Anscombe (IAT) para o realce das estruturas de interesse da imagem filtrada.

Transformada de Anscombe (AT) - A transformada de Anscombe é uma transformação não linear que permite transformar o ruído quântico de uma imagem digital, que é dependente do sinal, em um ruído aproximadamente independente do sinal, aditivo, Gaussiano, com média zero e variância unitária [Inouye, 1971; Romualdo, 2009].

De acordo com [Anscombe, 1948], dada a variável aleatória $\tilde{U}_i$, com distribuição estatística de Poisson, podemos expressar a AT desta variável aleatória, utilizando a equação (3.3)

$$\tilde{Z}_i = 2 \cdot \sqrt{\tilde{U}_i + \frac{3}{8}} \quad (3.3)$$

Esta nova variável $\tilde{Z}_i$, poder ser representada por meio de um modelo aditivo [Inouye, 1971], conforme mostra a equação (3.4)

$$\tilde{Z}_i = 2 \cdot \sqrt{\tilde{U}_i + \frac{1}{8}} + \tilde{N}_i = \tilde{S}_i + \tilde{N}_i \quad (3.4)$$

sendo $\tilde{N}_i$ um ruído aproximadamente independente do sinal $\tilde{S}_i$, descrito por uma distribuição Gaussiana com média zero e variância unitária.

Em outras modalidades de imagens médicas degradadas pelo ruído Poisson, a transformada de Anscombe tem proporcionado bons resultados para filtragem do ruído antes do processo de restauração e reconstrução de imagens [Mascarenhas et al., 1999; Homem et al., 2002; Homem et al., 2004; Romualdo, 2009]. Após esta transformação, podem-se utilizar técnicas bem conhecidas aplicadas à nova variável, para a redução do ruído, que agora é aditivo e independente do sinal [Mascarenhas et al., 1999].

Filtro de Wiener pontual (Lee) - Sob o critério do mínimo erro quadrático médio, o Filtro de Winer pontual [Lee, 1980] é um filtro linear ótimo e pode ser utilizado para a filtragem do ruído aditivo gaussiano. Assim, dado o vetor de observações u , para cada u_i , a Transformada de Anscombe é agora expressada pela equação (3.5)

$$z_i = 2 \cdot \sqrt{u_i + \frac{3}{8}} \quad (3.5)$$

Obtém-se uma estimativa sem ruído $\hat{s}_i$ para z_i utilizando o Filtro de Wiener pontual dado pela equação (3.6)

$$\hat{s}_i = E[\tilde{S}_i] + \frac{\sigma_{\tilde{s}_i}^2}{\sigma_{\tilde{s}_i}^2 + 1} (z_i - E[\tilde{S}_i]) \quad (3.6)$$

sendo que a média e a variância de $\hat{s}_i$, expressas respectivamente por $E[\tilde{S}_i]$ e $\sigma_{\tilde{s}_i}^2$, são medidas locais e podem ser estimadas na prática a partir da imagem a ser filtrada.

Transformada Inversa de Anscombe (IAT) - Após a filtragem do ruído, aplicamos a transformação inversa de Anscombe para termos uma estimativa da imagem degradada sem o ruído quântico [Romualdo, 2009]. Podemos expressar a IAT sobre $\hat{s}_i$ pela equação (3.7)

$$\hat{b}_i = \frac{1}{4} \cdot \hat{s}_i^2 - \frac{1}{8} \quad (3.7)$$

A Figura 13 ilustra o diagrama de processo para a filtragem de ruído Poisson utilizando a transformada de Anscombe junto com o filtro de Wiener pontual, para finalmente aplicar a transformada inversa de Anscombe.

Figura 13 - Ilustração do procedimento para redução do ruído Poisson utilizando a transformada de Anscombe (AT), o filtro de Wiener pontual e a Transformada Inversa de Anscombe (IAT).

3.5. Segmentação de Imagens

Baseados nos aspectos teóricos e nas justificativas apresentadas na secção 2.4 no capítulo anterior, agora vamos detalhar algumas definições com o objetivo de estabelecer os critérios que foram utilizados na implementação e execução dos algoritmos responsáveis de realizar as tarefas de segmentação neste projeto.

É importante indicar que um dos conceitos fundamentais da teoria Fuzzy Connectedness e do processo de segmentação é o termo de Afinidade, pelo fato de que através dela ficam estabelecidos os parâmetros com os quais a segmentação será feita. No entanto, também serve para delimitar suas limitações [Lage, 2010]. Em [Gonzalez; Woods, 2008], a Afinidade está conformada por três propriedades fundamentais: **adjacência** (μ_α), **homogeneidade** (μ_ψ), e **intensidade** (μ_ϕ). A afinidade, portanto, é uma função das propriedades anteriormente mencionadas:

$$\mu_k = \mu_\alpha(c, d) \cdot g(\mu_\psi(c, d), \mu_\phi(c, d)) \quad (3.8)$$

Sabe-se que a relação de adjacência $\mu_\alpha(c, d)$, indica a proximidade espacial entre os elementos avaliados, se c e d são vizinhos a adjacência Fuzzy é unitária. Com isso a afinidade fica expressada:

$$\mu_k = g(\mu_\psi(c, d), \mu_\phi(c, d)) \quad (3.9)$$

A definição do componente de homogeneidade $\mu_\psi(c, d)$ é dada pela seguinte expressão:

$$\mu_\psi(c, d) = \exp \left[-\frac{1}{2} \cdot \left(\frac{|f(c) - f(d)| - m_1}{s_1} \right)^2 \right] \quad (3.10)$$

na qual m_1 e s_1 são a média e o desvio padrão das homogeneidades locais do objeto, respectivamente. Assim mesmo, a definição do componente de intensidade $\mu_\phi(c, d)$ estaria dada por:

$$\mu_\phi(c, d) = \exp \left[-\frac{1}{2} \cdot \left(\frac{\left(\frac{f(c) + f(d)}{2} \right) - m_2}{s_2} \right)^2 \right] \quad (3.11)$$

onde m_2 e s_2 são a média e o desvio padrão das intensidades do objeto, respectivamente. Tanto na equação de homogeneidade quanto na equação de intensidade, $f(x)$ representa a intensidade do spel x .

Agora que são conhecidas as expressões tanto do componente de homogeneidade quanto de intensidade que integram a equação 3.9, pode-se definir finalmente o termo de afinidade da seguinte maneira:

$$\mu_k = w_1 \cdot \mu_\psi(c, d) + w_2 \cdot \mu_\phi(c, d) \quad (3.12)$$

onde w_1 e w_2 são valores numéricos não negativos que representam os pesos das componentes que conformam a expressão de afinidade, sendo que $w_1 + w_2 = 1$ [Nyúl et al., 2002].

Baseados nessas definições teóricas e complementadas com as relações globais de Conectividade Fuzzy entre os spels de uma imagem vistas na secção 2.4, seguidamente serão abordados os algoritmos que foram desenvolvidos e com os quais pretendemos resolver os assuntos de segmentação das imagens 3D que serão analisadas.

a) Algoritmo Genérico

Este algoritmo é o método de segmentação mais básico proposto em [Nyúl et al., 2002; Udupa; Saha, 2003]. Para isso, o primeiro passo é

definir uma semente (seed) que pertence ao objeto de interesse que se quer segmentar. A partir desse instante, são calculados os valores de adjacência, afinidade e conectividade de cada um dos spels da imagem com respeito à semente definida (segundo as equações definidas na secção 3.5). Estes valores são armazenados numa tabela e baseado numa relação ou grau de pertinência estabelecido, é definido quais dos elementos analisados pertencem e quais não ao objeto de interesse, colocando na imagem resultante valores igual a 1 para aqueles que conformem o objeto segmentado e 0 para o resto. Esta abordagem utiliza uma manipulação dos dados proposta por Dijkstra [Nyúl et al., 2002; Dijkstra, 1959].

É importante indicar que os pesos usados nesta primeira abordagem de segmentação, foram valores para w_1 e w_2 iguais a 0.5. No entanto, na secção 4.4.1 é feita uma análise que procura obter os pesos ótimos que consigam oferecer os melhores resultados.

b) Algoritmo LIFO

Esta abordagem mantém o enfoque do algoritmo descrito no item (a). No entanto, introduz uma importante variante baseada em adotar um método mais eficiente de administrar as estruturas de dados com os valores de cada um dos parâmetros envolvidos no processo de segmentação [Nyúl et al., 2002].

Especificamente neste caso, a manipulação dos valores armazenados na tabela, está regida pelo conceito de Dial [Dial, 1969] e utiliza a estrutura de dados LIFO (do inglês: last input–first output). Inicialmente, os valores de w_1 e w_2 foram iguais a 0.5, mas na secção 4.4.1 foram feitos testes que permitiram achar os melhores pesos.

c) Algoritmo kTetaFOEMS

Este algoritmo, consiste basicamente em identificar um objeto fuzzy $O_{K\theta}(s)$ definido em C , de tal forma que para uma semente qualquer $s \in S$, esse objeto cumpra as seguintes condições [Lage, 2010; Udupa; Saha, 2003]:

- $s \in O_{K\theta}(s)$
- $\forall c, d \in O_{K\theta}(s), \mu_K(c, d) \geq \theta$
- $\forall c \in O_{K\theta}(s) \text{ e } d \notin O_{K\theta}(s), \mu_K(c, d) < \theta$

Isso significa que tem que ser definido um valor limiar θ através do qual se estabeleça uma pertinência mínima, permitindo que um determinado spel forme parte do objeto de interesse [Udupa; Saha, 2003]. Finalmente como resultado da segmentação, é obtido um objeto Fuzzy $O_{K\theta}(s)$ em C .

Mais uma vez, os valores iniciais que tomaram tanto w_1 quanto w_2 foram iguais a 0.5.

d) Algoritmo Pesos Dinâmicos

Este algoritmo apresenta uma particularidade interessante em relação aos outros três primeiros algoritmos de segmentação. A particularidade desta abordagem consiste em realizar ajustes dinâmicos dos pesos da equação 3.12, que relacionam os componentes de homogeneidade e intensidade, para cada par de spels (c, d) de um caminho $p_{c,d}$ [Nyúl, 2002; Udupa; Saha, 2003]. Assim, os pesos adotados por w_1 e w_2 para cada um dos pares de elementos, c e d , que pertencem ao mesmo caminho vem expressado pelas seguintes equações [Lage, 2010]:

$$w_2 = \frac{\mu_\phi(c, d)}{\mu_\psi(c, d) + \mu_\phi(c, d)} \quad (3.13)$$

$$w_1 = 1 - w_2 \quad (3.14)$$

3.6. Avaliação dos resultados

Esta etapa de avaliação tem uma importância muito grande em nossa pesquisa, porque representa um processo que nos permite conhecer o nível de eficiência, acurácia, precisão e entender a significância de certos elementos dos algoritmos, tanto de filtragem quanto de segmentação, apresentados ao longo deste trabalho. Na literatura, são conhecidas e utilizadas muitas métricas para este fim. No entanto, seleccionou-se uma abordagem, cuja solução se apresenta mais completa em relação ao presente projeto, o qual é descrito em [Udupa et al., 2002].

Esta análise consiste basicamente em definir os parâmetros comparativos: Verdadeiro Positivo (TP) e Falso Positivo (FP), a fim de garantir um resultado confiável e sustentar a contribuição deste trabalho. Contudo, os parâmetros citados podem ser quantificados partindo da seguinte relação:

$$TP = \frac{A}{C} \quad (3.15)$$

$$FP = \frac{B_1 + B_2}{C} \quad (3.16)$$

sendo C a segmentação padrão (*gold standard*), A representa a intersecção da segmentação padrão com a segmentação obtida, B_1 e B_2 representam a diferença da segmentação padrão com a segmentação processada. Na Figura 14 pode-se representar a representação gráfica dos parâmetros TP e FP, utilizados neste processo.

Figura 14 – Ilustração geométrica dos fatores utilizados no processo de Avaliação.
[Udupa et al., 2002] .

Além destes dois parâmetros utilizados no processo de Avaliação, incorporou-se um terceiro parâmetro para dar maior consistência a esta etapa de avaliação. Trata-se da Máxima Distância (Max_{dist}) entre o contorno da imagem padrão ($Cont_p$) e o contorno da imagem segmentada ($Cont_s$) [Moraes; Furuie, 2011]. Este parâmetro nos permitirá ter uma noção do máximo desvio existente entre as duas imagens comparadas. Este parâmetro está representado pela seguinte equação:

$$Max_{dist} = \max|Cont_p - Cont_s| \quad (3.17)$$

3.7. Quantificação dos Volumes

O cálculo dos volumes dos objetos é uma tarefa simples e bastante direta, já que os objetos de interesse ficaram bem definidos após a etapa de segmentação. Este procedimento será realizado sobre o grupo de volumes PET reais, cujas aquisições foram geradas num período de tempo determinado.

4.1. Aquisição / Simulação de Imagens Médicas 3D

Foram utilizados dois tipos de imagens tridimensionais simuladas que serviram de entrada na sequência de processamento implementada. No início, foi preciso gerar um volume simples, mas com todas as características necessárias das imagens de Medicina Nuclear da modalidade PET reais, nesse sentido tendo em consideração os pontos mencionados na seção 3.3. (b) foi desenvolvido o volume simulado (*phantom*) do Ventrículo Esquerdo do coração. Posteriormente, com o objetivo de avaliar mais estritamente nossa sequência de processamento, utilizamos *phantoms* simulados com NCAT-4D, os quais representam imagens de entrada mais próximas às imagens reais. Adicionalmente, para encerrar nossa análise, foram utilizadas e testadas imagens reais na sequência de processamento que foi implementado.

4.1.1. Ventrículo Esquerdo simulado

Através de um algoritmo elaborado em MatLab, foi possível simular a cavidade do Ventrículo Esquerdo do coração, tomando em consideração as dimensões e a espessura da parede da cavidade, mencionadas na literatura para este fim [Fifer; Grossman, 2000; Wikipedia, 2011]. Na Figura 16, apresenta-se uma representação bastante simples do volume da cavidade do Ventrículo Esquerdo simulado.

Para ter uma melhor concepção do desenvolvimento deste *phantom* listamos na Tabela 3 os valores dos principais parâmetros considerados na elaboração do Ventrículo Esquerdo simulado. Este conjunto de valores específicos forma parte da simulação da cavidade do Ventrículo Esquerdo de uma pessoa adulta qualquer (como exemplo, do primeiro paciente: Caso_H01 do grupo de Homens Adultos apresentado no Anexo A), onde inicialmente definimos as dimensões do volume que

conterá o Ventrículo Esquerdo simulado, de 128x128x128 pixels (largura, altura e profundidade do volume). Em seguida são especificados os parâmetros que definirão o volume simulado do Ventrículo Esquerdo propriamente dito, para o qual é necessário ingressar a distância (em pixels) das coordenadas x , y e z , já que este processo parte da utilização da fórmula da parábola girada 360 graus sobre o eixo vertical (eixo z), assim como a determinação da espessura da parede. Além disso, é possível definir tanto o valor das intensidades do Ventrículo Esquerdo (objeto) e do background, quanto a distância dos cortes entre cada fatia gerada. O algoritmo de simulação do VE pode ser revisado no Anexo G.

Ventrículo Esquerdo Simulado (Parâmetros)	
Descrição	Valor
Largura do volume	128 pixels
Altura do volume	128 pixels
Profundidade do volume	128 pixels
Distância máxima em x	60 pixels
Distância máxima em y	60 pixels
Distância máxima em z	80 pixels
Espessura da parede	12 pixels
Intensidade do Background	2
Intensidade do Objeto	20
Distância dos cortes	1 pixel
Total fatias geradas	128

TABELA 3 – Principais parâmetros considerados no desenvolvimento do Ventrículo Esquerdo Simulado.

No entanto, para nosso projeto o interesse é gerar um banco de volumes a fim de testar nossos algoritmos com cada um deles. Para isso, foram mantidas as dimensões da matriz 3D (128x128x128) e fomos alterando os parâmetros, tais como o tamanho do ventrículo esquerdo (largura, altura e espessura da parede), os valores dos níveis de cinza (intensidade), tanto do objeto quanto do background, e o ruído Poisson, o qual é aleatoriamente incorporado em cada uma das fatias que compõem cada volume simulado. Deste modo, conseguimos simular 30 volumes divididos em três grupos da seguinte maneira: o primeiro grupo

conformado por volumes pertencentes a 10 homens adultos, o segundo grupo conformado por volumes pertencentes a 10 mulheres adultas e finalmente o terceiro grupo conformado por 10 volumes de crianças.

FIGURA 16 – Simulação simples do Ventriculo Esquerdo do coração. (Imagem das partes do coração adaptada de <http://campus.fortunecity.com/>).

No Anexo A pode-se visualizar os valores dos parâmetros estabelecidos para cada um dos 30 volumes simulados.

4.1.2. *Phantoms* simulados com o NCAT-4D

Foram proporcionados pelo Serviço de Informática do Instituto do Coração (InCor) do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo (HC-FMUSP) alguns arquivos de *phantoms* simulados com NCAT-4D, os quais serão utilizados como imagens de entrada da nossa sequência de processamento. Para ter uma idéia mais clara deste tipo de *phantoms*, são mostradas na Figura 17 algumas imagens que simulam o fígado com o coração (nome do arquivo: liver+heart60bpm_act_TOTAL.bin). Já na Figura 18 é mostrada a

simulação do Ventrículo Esquerdo do Coração junto com estruturas da caixa torácica (nome do arquivo: ncat_lv60bpm_act_TOTAL.bin). No Anexo B são apresentados em detalhe os principais parâmetros deste último *phantom*, que será objeto de estudo no projeto.

FIGURA 17 – *Phantom* simulado com o uso do NCAT-4D. (a) Conjunto de fatias 2D das projeções dos órgão pelo eixo z, (b) Vista frontal do volume gerado com todas as fatias usando o ImageJ.

FIGURA 18 – *Phantom* simulado com o uso do NCAT-4D. (a) fatias 2D das projeções do ventrículo esquerdo dentro da caixa torácica, (b) volume gerado com ajuda do ImageJ.

4.1.3. Imagens PET reais

Conforme mencionado na seção 3.3, tivemos acesso ao arquivo de um exame PET real, o qual foi realizado num paciente de sexo masculino. O referido exame está constituído por um conjunto de 36 volumes, adquiridas em períodos de 300 segundos cada. A figura 19 mostra por um

lado a sequência de algumas das fatias do miocárdio estudado e por outro o volume 3D reconstruído. No Anexo C são apresentados os dados mais relevantes deste exame PET real, assim como uma descrição detalhada dos passos para o processo de reconstrução deste grupo de volumes PET reais.

FIGURA 19 – Exame PET real. (a) fatias 2D das projeções das cavidades do miocárdio, (b) volume gerado por meio de ferramentas de reconstrução do ImageJ.

4.2. Adição do Ruído quântico tipo Poisson

As imagens, tanto do VE simulado em MatLab quanto daquele obtido através do NCAT-4D, não contém nenhum tipo de ruído. Como o nosso objetivo é avaliar imagens próximas às imagens reais, é vital que as imagens inicialmente adquiridas adotem as principais características de uma imagem PET real. Tal como foi explicado na secção 3.4, é preciso incorporar ruído Poisson sobre todas as imagens que conformam nosso volume de entrada. Para isso, foi desenvolvido um algoritmo em MatLab (através da função: `imnoise[image,'Poisson']`), através do qual foi possível incorporar aleatoriamente o ruído Poisson próprio das imagens de Medicina Nuclear na modalidade PET. Este procedimento foi aplicado em cada uma das fatias que compõem os volumes simulados (*phantom* próprio e aquele obtido por meio do NCAT-4D).

Um exemplo desta incorporação é mostrado na Figura 20, onde se pode ver que o *phantom* em (a) não contém ruído nenhum, já em (b) foi adicionado o ruído tipo Poisson. A implementação deste algoritmo de incorporação de ruído Poisson pode ser visualizada no Anexo G.

FIGURA 20 – (a) fatia 2D de uma secção do ventrículo esquerdo dentro da caixa torácica, (b) fatia 2D com ruído Poisson.

4.2.1. Avaliação do Ruído Poisson

Uma vez incorporado o ruído Poisson nos volumes que na sequência passarão pelos outros elementos (filtros, segmentadores, avaliadores, quantificadores) que fazem parte da estrutura de processamento implementada, surge uma questão válida e interessante neste ponto: Será que realmente o ruído que foi incorporado corresponde ao ruído Poisson requerido? Para comprovar este procedimento de incorporação do ruído Poisson, vamos aplicar alguns testes estatísticos para confirmar e validar esta suposição.

Este método de avaliação consiste na utilização de um conceito estatístico bastante utilizado e conhecido, chamado ‘*Chi-Square Goodness-of-Fit Test*’ [Nist/Sematech, 2011]. Para isso, o primeiro passo consistiu em gerar volumes com as mesmas dimensões (128 x 128 x 128 voxels), mas com diferentes valores de intensidade, sendo os valores escolhidos os seguintes: 5, 10, 20, 30, 40 e 50, para cada um dos

volumes gerados. Logo após foi incorporado o ruído Poisson, mediante o procedimento explicado na secção 4.2.

A seguir, estabelecemos nossa hipótese que considera que o ruído incorporado, através de um processo que seguiu uma distribuição de probabilidade Poisson, é efetivamente ruído Poisson.

Uma das coisas que esperamos observar é que numa distribuição de Poisson, a média e a variância devem ser iguais ou aproximadamente iguais. Desta forma, será necessário corroborar a semelhança destes valores, ou de modo análogo que a relação $\text{Var}(x)/E(x)$ deve ser próximo a 1, para determinar que o ruído analisado efetivamente corresponde ao ruído Poisson [Abe, 1999].

O teste pelo método do *Qui-quadrado Goodness-of-Fit Test* consiste em obter uma relação entre os valores de frequências obtidas através dos histogramas dos volumes, comparados com os valores esperados (teóricos) obtidos dos cálculos, a partir da fórmula da distribuição de probabilidade Poisson apresentada na secção 3.4. (equação 3.1).

O objetivo é calcular o valor do qui-quadrado, segundo a seguinte relação entre os valores das frequências observadas e esperadas mencionadas anteriormente:

$$X^2 = \sum_{i=1}^k \frac{(\text{observado}_i - \text{esperado}_i)^2}{\text{esperado}_i} = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i} \quad (4.1)$$

$$E_i = N[F(Y_u) - F(Y_l)] \quad (4.2)$$

onde O_i representa as frequências observadas, E_i as frequências esperadas, k o número total de classes, N o tamanho da amostra, F é a função distribuição de probabilidade acumulada, Y_u o limite superior para a classe i e Y_l o limite inferior para a classe i .

Portanto, uma vez calculados os valores qui-quadrado (X^2) baseados nos dados experimentais e teóricos obtidos apriori, temos dois caminhos:

- Por um lado, se $X^2 \leq \chi^2(1 - \alpha; k - m - 1) \rightarrow H_0$: A distribuição é Poisson ou dito de outra maneira a concordância experimental corresponde à distribuição especificada [Nist/Sematech, 2011].
- Caso contrário, se $X^2 > \chi^2(1 - \alpha; k - m - 1) \rightarrow H_1$: A distribuição não é Poisson ou apresenta uma concordância experimental progressivamente fraca. Nesse caso, a hipótese de que os dados são de uma população com distribuição especificada, em nosso caso do tipo Poisson é rejeitada [Nist/Sematech, 2011].

Para o teste foram considerados valores teóricos com 0,05 (5%) de nível de significância, segundo a tabela teórica da distribuição qui-quadrado do Anexo D.

Para os valores examinados, concluímos que para os valores de intensidade igual a 5, 10, 20, 30, 40 o teste determinou que a distribuição especificada corresponde à distribuição Poisson. Portanto, não se rejeita a hipótese de ruído Poisson.

No entanto, para o valor de intensidade igual a 50 (como para valores maiores de intensidade), foi rejeitada a hipótese inicial, com o qual foi possível julgar que o ruído analisado não se tratava do ruído Poisson desejado. Com o veredito da análise, quisemos averiguar se o ruído com valor de intensidade igual a 50, que não é Poisson, corresponde ao ruído Gaussiano. Para isso, aplicamos a mesma metodologia com a variante que agora os valores de frequência esperada (teóricos) foram produto da aplicação da fórmula de distribuição Gaussiana. Finalmente, foi factível determinar que o ruído com valor de intensidade igual a 50 corresponde ao ruído Gaussiano.

Os valores quantitativos desta avaliação são mostrados nas Tabelas 23, 24, 25, 26, 27, 28 e 29 no Anexo D. Assim, pode-se

determinar que o ruído incorporado nos volumes do VE simulado que foram utilizados na sequência de processamento neste projeto, corresponde ao ruído Poisson requerido já que os valores de intensidade destes volumes estão dentro das intensidades avaliadas como sendo hipóteses verdadeiras. No entanto, para o volume do VE gerado através do NCAT-4D, o qual tem um valor de intensidade igual a 100 como é mostrado no Anexo B, não se pode considerar que o ruído incorporado corresponde ao ruído Poisson requerido.

4.3. Pre-processamento: Filtragem das imagens 3D

Um dos passos mais relevantes antes do processo de segmentação é a parte da filtragem, já que o objetivo é tirar da melhor maneira possível todas as “imperfeições” contidas na imagem, que de alguma forma comprometem a informação importante do objeto de interesse. Tal como foi detalhado na secção 3.4, foram implementados três diferentes abordagens para a filtragem das imagens 3D.

FIGURA 21 – (a) Imagem de uma fatia do VE simulado, (b) fatia do VE simulado com ruído Poisson incorporado e (c) resultados qualitativos do filtro da mediana sobre o ruído Poisson com três tamanhos de janelas diferentes. Foram alterados os níveis de contraste e brilho das imagens mostradas, mudando a escala de intensidades para melhor visualização.

O filtro da Mediana foi aplicado sob o volume do VE simulado e sob o *phantom* do VE originado com o NCAT-4D, partindo de três tamanhos de janela ou máscara diferentes: 3×3 , 5×5 e 7×7 , obtendo-se os resultados mostrados nas figuras 21 e 22, respectivamente.

FIGURA 22 – (a) Imagem de uma fatia do *phantom* gerado através do NCAT-4D, (b) fatia com ruído Poisson incorporado e (c) resultados qualitativos do filtro da mediana sobre o ruído Poisson com três tamanhos de janelas diferente.

Assim mesmo, usando o filtro da Gaussiana ponderada foram feitos testes com diferentes valores de “sigma” tanto para o VE simulado, quanto para o *phantom* gerado pelo NCAT-4D. Os valores sigma testados foram os seguintes: $\sigma = 0.5$, $\sigma = 1.0$, $\sigma = 1.5$, $\sigma = 2.0$, $\sigma = 2.5$ e $\sigma = 3.0$, obtendo-se os resultados mostrados nas figuras 23 e 24.

FIGURA 23 – Resultados qualitativos do filtro da Gaussiana ponderada sobre o ruído Poisson do VE simulado, usando diferentes valores de sigma. Foram alterados os níveis de contraste e brilho das imagens mostradas, mudando a escala de intensidades para melhor visualização.

FIGURA 24 – Resultados qualitativos do filtro da Gaussiana ponderada sobre o ruído Poisson do *phantom* gerado por meio do NCAT-4D, usando diferentes valores de sigma.

Finalmente, foi testada a abordagem de filtragem, utilizando a Transformada de Anscombe junto com o Filtro de Wiener pontual. Na Figura 25, se mostram duas fatias bidimensionais similares de uma

porção do Ventrículo Esquerdo simulado de acordo ao indicado em 4.1. A primeira fatia representada em (a) contém ruído quântico tipo Poisson e a outra fatia representada em (b), é a fatia resultante depois da execução do filtro.

Se percorrermos uma linha central e horizontal em cada uma das duas imagens e analisarmos o perfil de cada uma delas, utilizando a função <Profile> do ImageJ para este fim, teremos como resultante os gráficos apresentados a continuação, onde claramente pode-se distinguir a considerável eliminação do ruído Poisson.

O mesmo procedimento foi realizado no conjunto de fatias que confirmam os volumes dos *phantoms* gerados com o NCAT-4D e para os volumes PET reais que foram utilizados no projeto.

FIGURA 25 – Perfil (linha central horizontal) de (a) Imagem com ruído Poisson e (b) Imagem filtrada.

Nas Tabelas 4 e 5 da secção 4.6 Avaliação dos Resultados, são apresentados os resultados quantitativos obtidos ao final da sequência de

processamento, tendo usando as três abordagens de filtragem que foram apresentadas na secção 3.4. Estes testes foram aplicados num volume do VE simulado, assim como no *phantom* do VE gerado com o NCAT-4D. O que procuramos conhecer foi qual dos três filtros em prova oferecia um melhor desempenho dentro da sequência de processamento implementada e ao mesmo tempo qual permitia obter o resultado ótimo dentro do objetivo proposto pelo projeto. Com esse propósito, foi possível determinar também o tamanho da janela no caso do filtro da mediana e o valor de sigma do filtro da gaussiana ponderada, que providenciavam os melhores resultados.

Estes parâmetros do tamanho da janela e do valor de sigma dos filtros da mediana e gaussiana ponderada, respectivamente, que representaram os melhores resultados nas Tabelas 4 e 5 serão utilizados posteriormente para testar o grupo de imagens PET reais.

4.4. Segmentação de estruturas de interesse nas imagens 3D

Os quatro algoritmos abordados possuem suas formulações iniciais descritas em [Nyúl et al., 2002; Udupa; Saha, 2003]. Estes algoritmos foram escolhidos por sua simplicidade, robustez e bom desempenho na tarefa de segmentação que estamos querendo investigar.

A Figura 26 mostra a segmentação do *phantom* 3D gerado com o NCAT-4D, por meio do algoritmo com Pesos Dinâmicos. Pode-se verificar que as imagens resultantes unicamente correspondem ao objeto de interesse selecionado, que neste caso é o Ventrículo Esquerdo. A particularidade e grande vantagem do algoritmo com Pesos Dinâmicos, como foi explicada na secção 3.5(d), é que realiza um processo de escolha dos melhores pesos w_1 e w_2 , os quais participam diretamente no cômputo do valor de afinidade, conforme foi indicado nas equações 3.13 e

3.14. Já nos outros três primeiros algoritmos, os valores das constantes w_1 e w_2 foram fixados num valor igual a 0,5 para cada um deles.

FIGURA 26 – *Phantom* do ventrículo esquerdo simulado com o uso do NCAT-4D.

(a) Conjunto de fatias 2D das projeções do VE junto com as estruturas da caixa torácica,

(b) Fatias 2D obtidas depois do processo de Segmentação, utilizando o algoritmo Conectividade Fuzzy com Pesos Dinâmicos.

4.4.1. Curvas Características de Operação do Receptor (ROC)

A Curva ROC é uma ferramenta poderosa que permite medir e especificar problemas no desempenho do diagnóstico em medicina, por permitir estudar a variação da sensibilidade e especificidade para diferentes valores de corte. Esquemáticamente, a Curva ROC é um gráfico de sensibilidade que representa a taxa de verdadeiros positivos versus a taxa de falsos positivos [Margotto, 2010; Braga, 2003].

A Curva ROC permite evidenciar os valores para os quais existe maior otimização da sensibilidade em função da especificidade, que corresponde ao ponto em que se encontra mais próxima do canto superior esquerdo do diagrama, uma vez que o índice de verdadeiros positivos é 1 e o de falsos positivos é zero.

Esta ferramenta será usada para determinar os valores dos pesos w_1 e w_2 que pertencem às equações 3.13 e 3.14, empregada para o cálculo do valor de afinidade por meio da equação 3.12 no processo de

segmentação. Em [Nyul et al., 2002] foram testados alguns valores para estas constantes, sendo por um lado $w_1 = 1/3$ e $w_2 = 2/3$ e por outro $w_1 = 2/3$ e $w_2 = 1/3$. Para o nosso estudo de avaliação, foram estendidos os valores atribuídos a w_1 e w_2 com o objetivo de achar os pesos que consequentemente se reflitam numa segmentação ótima e com melhores resultados.

Os pares de valores testados para w_1 e w_2 , apresentados nas Tabelas 31, 32 e 33 no Anexo E, são: $w_1 = 0.9$ e $w_2 = 0.1$, $w_1 = 0.8$ e $w_2 = 0.2$, $w_1 = 0.7$ e $w_2 = 0.3$, $w_1 = 0.6$ e $w_2 = 0.4$, $w_1 = 0.5$ e $w_2 = 0.5$, $w_1 = 0.4$ e $w_2 = 0.6$, $w_1 = 0.3$ e $w_2 = 0.7$, $w_1 = 0.2$ e $w_2 = 0.8$, $w_1 = 0.1$ e $w_2 = 0.9$. Estes testes foram aplicados no processo de segmentação de um dos volumes simulados do VE usando os algoritmos Genérico, LIFO e kTetaFOEMS com valor teta igual a 0.10 ($\theta = 0.10$).

Depois de feitas as provas, foi determinado que para valores de $w_1 = 0.3$ e $w_2 = 0.7$, a segmentação das imagens filtradas (sequência completa de processamento) apresentou melhores resultados quando quantificado através dos algoritmos de avaliação. Tendo determinado os melhores pesos de w_1 e w_2 para o processo de segmentação, estes valores serão utilizados para a segmentação dos volumes PET reais no processo de quantificação dos volumes.

4.5. Pós filtragem através de morfologia matemática

Já que os resultados em termos gerais estão sendo satisfatórios de acordo com nossas expectativas, não temos necessidade de implementar uma etapa de pós filtragem. No entanto, se posteriormente testarmos algumas outras imagens de entrada (*phantoms* ou imagens reais) e os resultados não forem os desejados, logicamente será desenvolvida e implementada esta fase.

4.6. Avaliação dos resultados

Depois de ter feito toda nossa sequência de passos desde a geração das imagens padrões, passando pela incorporação de ruído Poisson, filtragem e segmentação, será feito o procedimento de avaliação através da contagem e cálculo da porcentagem de Verdadeiro Positivo (VP) e Falso Positivo (FP), comparando as imagens padrões e/ou gold standard, com cada uma das imagens obtidas como resultado da sequência de processamento em distintos pontos: (i) imagens inicialmente simuladas, (ii) imagens com ruído Poisson incorporado, (iii) imagens após a etapa de filtragem e (iv) imagens segmentadas.

O algoritmo proposto solicita ao usuário que especifique o diretório onde estão armazenadas as imagens padrão ou gold standard, e o diretório com as imagens que foram processadas (filtradas e/ou segmentadas). A seguir, faz-se a comparação de cada par de fatias, de acordo com os nomes dos arquivos de ambos diretórios escolhidos. Um a um se realizam os cálculos das áreas da imagem padrão (região C), a intersecção (região A) e a diferença (região B) das imagens analisadas, segundo o esquema mostrado na Figura 14. Com estes valores, usando as equações 3.15 e 3.16 obtemos os resultados dos dois primeiros parâmetros de avaliação, que são VP e FP, respectivamente.

Na sequência, para calcular a Máxima Distância (Max_{dist}) como o terceiro parâmetro de avaliação, gerou-se uma imagem do contorno (borda), tanto da imagem padrão quanto da imagem processada e fazemos uma comparação (pixel a pixel) da diferença entre as imagens, obtendo a máxima distância ou o máximo desvio existente entre uma imagem com relação à outra.

No caso das imagens simuladas do VE, as imagens padrões serão as imagens geradas no processo inicial (antes da incorporação do ruído Poisson). Já para os *phantoms* gerados pelo NCAT-4D, uma vez que a imagem original representava o VE junto com a caixa torácica, uma etapa

intermediária de segmentação (por meio de limiarização) foi realizada para capturar unicamente o objeto de estudo, que é o VE. O produto desta operação será nossa imagem gold standard que será comparada com as imagens processadas por meio do procedimento de avaliação definido.

No transcurso do projeto foram incorporadas novas abordagens de filtragem, tal foi detalhado nas secções 3.4 e 4.3. Cada um dos algoritmos testados apresentou um conjunto de variantes (parâmetros que definiram o nível de filtragem), pelo que foi necessário fazer uma análise completa tanto sobre o volume do VE simulado quanto sobre o *phantom* gerado por meio do NCAT-4D. As Tabelas 4 e 5 mostram os resultados destes testes, aplicados sobre os dois conjuntos de dados utilizados, respectivamente; usando todas as variáveis tanto dos algoritmos de filtragem como dos algoritmos de segmentação. Cabe ressaltar que, no caso do Filtro da Mediana, a filtragem usando janela 5 x 5 apresentou melhores resultados comparado com os outros dois testes para ambos volumes simulados. No caso do Filtro da Gaussiana Ponderada, através de um valor de sigma igual a 1.5 foi possível conseguir resultados ótimos comparado com os outros cinco valores de sigma diferentes.

FILTROS		SEGMENTADOR							
		Genérico		LIFO		Kteta (0.10)		Pesos Dinâmicos	
		TP %	FP %	TP %	FP %	TP %	FP %	TP %	FP %
Mediana	3 x 3	94.35	5.01	94.66	4.75	94.91	4.28	95.43	4.08
	5 x 5	95.12	4.73	95.32	4.52	95.86	4.13	96.77	3.97
	7 x 7	93.18	5.88	93.22	5.67	93.71	5.34	94.12	5.22
Gaussiana Ponderada	Sigma 0.5	92.72	5.98	92.84	5.71	93.23	5.46	93.56	5.27
	Sigma 1.0	94.55	4.01	94.80	4.54	94.91	4.28	95.33	3.89
	Sigma 1.5	95.45	4.25	95.71	4.22	95.80	4.13	96.13	3.75
	Sigma 2.0	94.23	4.47	94.35	4.72	94.88	4.31	95.02	4.02
	Sigma 2.5	93.78	5.73	93.92	5.52	94.11	5.13	94.30	4.79
	Sigma 3.0	93.34	6.12	93.76	6.06	93.24	5.73	93.77	5.59
Anscombe / Lee		97.10	2.72	97.76	2.32	97.88	2.78	97.93	2.31

TABELA 4 – Quadro comparativo dos resultados dos algoritmos de filtragem vs. segmentação para o volume do VE simulado.

FILTROS		SEGMENTADOR							
		Genérico		LIFO		Kteta (0.10)		Pesos Dinâmicos	
		TP %	FP %	TP %	FP %	TP %	FP %	TP %	FP %
Mediana	3 x 3	90.39	4.81	90.53	4.63	90.87	4.19	91.24	4.02
	5 x 5	91.11	3.63	91.43	3.58	91.58	3.81	91.92	3.45
	7 x 7	88.48	5.92	88.52	5.70	88.92	5.35	89.33	5.11
Gaussiana Ponderada	Sigma 0.5	88.56	5.85	88.75	5.77	89.22	5.44	89.37	5.30
	Sigma 1.0	89.88	4.30	90.04	4.24	90.34	4.28	90.77	4.12
	Sigma 1.5	91.45	3.32	91.71	3.02	91.80	3.00	92.23	2.91
	Sigma 2.0	89.49	4.74	89.62	4.24	89.87	4.17	90.22	4.12
	Sigma 2.5	88.74	5.79	88.89	5.72	89.35	5.48	89.59	5.21
	Sigma 3.0	88.08	6.22	88.46	5.96	88.86	5.43	89.26	5.37
Anscombe / Lee		92.12	2.51	92.58	2.11	93.09	2.23	94.05	2.31

TABELA 5 – Quadro comparativo dos resultados dos algoritmos de filtragem vs. segmentação para o *phantom* NCAT-4D.

No Capítulo 5 são apresentados os quadros resumo com os valores dos parâmetros (VP, FP e Max_{dist}) de avaliação, obtidos para os três grupos analisados e para um volume gerado pelo NCAT-4D.

No caso das imagens reais, não foi possível aplicar os algoritmos de avaliação ao não contar com as imagens gold standard. Neste caso em particular, foi feita numa primeira etapa a quantificação dos voxels de cada volume analisado; depois o resultado obtido foi comparado com dados teóricos de volumes do miocárdio (em mm^3). Os detalhes serão apresentados a seguir e complementados com dados no Anexo C.

4.7. Quantificação dos Volumes

O processo de quantificação consiste em contabilizar cada um dos voxels que pertencem ao objeto de interesse em cada um das fatias ou cortes que conformam o volume analisado. Para o caso das imagens reais, elas estão conformadas por um conjunto de 35 cortes ou fatias cada. Sendo imagens DICOM, entendendo por imagem DICOM aquela imagem radiológica que possui uma formatação padrão global que

permite uma transmissão de informação médica num formato eletrônico estruturando um protocolo definido, foi possível conhecer a dimensão exata de cada voxel, o qual multiplicado pela distância entre os cortes permitiu obter o volume do objeto representado, tanto em unidades de voxels quanto em mm^3 .

Cada um dos volumes reais analisados está constituído por 35 fatias de 128×128 que equivalem a $220 \times 220 \text{ mm}$. Fazendo os cálculos, obtém-se que cada pixel é igual a 2.95 mm^2 . Já que o espaçamento entre cada um dos 35 cortes que conformam os volumes PET é de 4.25 mm , fica determinado que cada voxel equivale a 12.55 mm^3 . Para mais detalhes respeito às imagens PET reais usadas no projeto, consultar o Anexo C.

Dos 36 volumes que contem o exame PET real, para propósitos do projeto serão usados unicamente os quatro últimos volumes. Como exemplo do procedimento seguido, na Tabela 6 são mostrados os resultados do processo de quantificação do último volume (volume N° 36) que é similar ao volume apresentado na Figura 19 da seção 4.1.3., onde se testaram todas as abordagens implementadas neste projeto, tanto de filtragem quanto de segmentação, com o fim de sustentar e/ou corroborar as escolhas feitas nas seções 4.3 e 4.6, respectivamente.

Segmentador	Filtro da Mediana			Filtro da Gaussiana Ponderada						Filtro de Anscombe/Lee
	3 x 3	5 x 5	7 x 7	Sigma 0.5	Sigma 1.0	Sigma 1.5	Sigma 2.0	Sigma 2.5	Sigma 3.0	
Genérico	6016	6620	3927	6680	7141	6939	5381	3601	1980	7355
LIFO	6974	7527	4522	7355	7564	7039	5685	3734	2065	7669
KTetaFOEMS (0.10)	7431	8028	4849	7669	8138	8574	6292	3817	2225	9522
Pesos Dinâmicos	8469	9915	6724	8369	9748	9849	7557	5209	3368	10257

TABELA 6 – Quadro comparativo dos resultados de quantificação (expressados em pixels) do volume PET N° 36.

Pode-se apreciar na tabela anterior (Tabela 6), como foi indicado quando foram analisadas as imagens do VE simulado e do *phantom*

gerado através do NCAT-4D, o seguinte: das abordagens de filtragem implementadas, o filtro de Anscombe/Wiener tem supremacia sobre os outros dois filtros testados, já no caso das abordagens de segmentação o algoritmo que usa Pesos Dinâmicos tem uma considerável vantagem com respeito aos outros três algoritmos. Além disso, pode-se verificar que para o caso do Filtro da Mediana, a janela 5 x 5 e para o Filtro da Gaussiana Ponderada com o valor de sigma igual a 1.5, fornecem os melhores resultados para propósitos deste projeto e serão os valores utilizados no análise dos outros volumes PET reais.

Da tabela 6, selecionamos o valor de quantificação que relaciona o melhor filtro com o melhor segmentador que será o valor representativo para este volume. Este valor de quantificação representativo, 10257 pixels para o caso do volume N° 36, corresponde à somatória dos elementos das 35 fatias que conformam este volume. No entanto, essa quantidade não corresponde ainda ao valor do volume total, já que temos que incorporar o valor de espaçamento entre os cortes. Usando os dados da secção 3.6 e do Anexo C temos que: 128 spels de um corte qualquer equivalem a 220 mm, então 1 spel seria igual a 1.71875 mm. Sendo que a distância entre os cortes é de 4.25 mm, isso equivale a 2.47 ou 2.5 spels de espaçamento. Então temos que:

$$\text{Volume N° 36} = 10257 \text{ pixels} * 2.5 \text{ spels}$$

$$\boxed{\text{Volume N° 36} \cong 25642 \text{ voxels}}$$

$$\text{Volume N° 36} = 10257 \text{ pixels} * 2.95 \frac{\text{mm}^2}{\text{pixels}} * 4.25 \text{ mm}$$

$$\boxed{\text{Volume N° 36} = 128597.14 \text{ mm}^3}$$

O consolidado dos valores dos quatro volumes analisados será apresentado no capítulo dos resultados na secção 5.5.

Capítulo 5

RESULTADOS

Em continuação, são apresentados os resultados de avaliação obtidos durante toda a sequência de processamento detalhada ao longo do Capítulo 4. Estes resultados consideram cada um dos 30 volumes pertencentes aos três grupos (homens, mulheres e crianças), o volume gerado pelo simulador NCAT-4D, além do grupo de imagens PET reais. Para maiores detalhes, consultar os Anexos A, B, C, D, E e F.

5.1 VE simulado: Imagens Padrões

As Tabelas 7, 8 e 9 mostram os parâmetros de avaliação no processo de segmentação das imagens Padrões, para cada um dos grupos inicialmente estabelecidos.

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.10$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
HOMENS	1	99.93	0.00	1	100.00	0.00	0	99.99	0.00	1	99.98	0.00	1
	2	99.12	0.00	1	99.61	0.00	1	100.00	0.00	0	99.96	0.00	1
	3	100.00	0.00	0	100.00	0.00	0	99.85	0.00	1	100.00	0.00	0
	4	100.00	0.00	0	99.50	0.00	1	100.00	0.00	0	100.00	0.00	0
	5	99.99	0.00	1	100.00	0.00	0	100.00	0.00	0	100.00	0.00	0
	6	100.00	0.00	0	100.00	0.00	0	99.79	0.00	1	99.84	0.00	1
	7	99.54	0.00	1	99.81	0.00	1	100.00	0.00	0	99.98	0.00	1
	8	100.00	0.00	0	99.95	0.00	1	99.68	0.00	1	100.00	0.00	0
	9	99.77	0.00	1	100.00	0.00	0	99.66	0.00	1	100.00	0.00	0
	10	99.63	0.00	1	100.00	0.00	0	100.00	0.00	0	99.98	0.00	1
Media		99.80	0.00	1	99.89	0.00	1	99.90	0.00	1	99.97	0.00	1
Desv. Padrão		0.29	0.00		0.19	0.00		0.14	0.00		0.05	0.00	

TABELA 7 – Parâmetros de Avaliação no processo de Segmentação das imagens Padrões para o grupo de Homens (10 volumes).

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.10$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
MULHERES	1	99.89	0.00	1	99.78	0.00	1	99.93	0.00	1	100.00	0.00	0
	2	99.65	0.00	1	100.00	0.00	0	100.00	0.00	0	99.85	0.00	1
	3	100.00	0.00	0	100.00	0.00	0	100.00	0.00	0	100.00	0.00	0
	4	100.00	0.00	0	99.62	0.00	1	99.86	0.00	1	100.00	0.00	0
	5	99.65	0.00	1	99.89	0.00	1	100.00	0.00	0	100.00	0.00	0
	6	100.00	0.00	0	100.00	0.00	0	99.79	0.00	1	99.97	0.00	1
	7	99.53	0.00	1	100.00	0.00	0	100.00	0.00	0	99.97	0.00	1
	8	99.76	0.00	1	99.78	0.00	1	100.00	0.00	0	99.98	0.00	1
	9	99.71	0.00	1	99.71	0.00	1	99.87	0.00	1	100.00	0.00	0
	10	100.00	0.00	0	100.00	0.00	0	99.98	0.00	1	100.00	0.00	0
Media		99.82	0.00	1	99.88	0.00	1	99.94	0.00	1	99.98	0.00	1
Desv. Padrão		0.18	0.00		0.14	0.00		0.08	0.00		0.05	0.00	

TABELA 8 – Parâmetros de Avaliação no processo de Segmentação das imagens Padrões para o grupo de Mulheres (10 volumes).

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.10$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
CRIANÇAS	1	100.00	0.00	0	99.53	0.00	1	100.00	0.00	0	99.93	0.00	1
	2	99.13	0.00	1	100.00	0.00	0	99.86	0.00	1	100.00	0.00	0
	3	100.00	0.00	0	100.00	0.00	0	100.00	0.00	0	99.89	0.00	1
	4	100.00	0.00	0	99.79	0.00	1	100.00	0.00	0	100.00	0.00	0
	5	100.00	0.00	0	100.00	0.00	0	99.84	0.00	1	100.00	0.00	0
	6	99.65	0.00	1	99.85	0.00	1	99.88	0.00	1	100.00	0.00	0
	7	99.83	0.00	1	100.00	0.00	0	100.00	0.00	0	99.90	0.00	1
	8	100.00	0.00	0	99.87	0.00	1	100.00	0.00	0	100.00	0.00	0
	9	99.76	0.00	1	100.00	0.00	0	99.91	0.00	1	99.96	0.00	1
	10	99.73	0.00	1	99.92	0.00	1	100.00	0.00	0	99.99	0.00	1
Media		99.81	0.00	1	99.90	0.00	1	99.95	0.00	1	99.97	0.00	1
Desv. Padrão		0.27	0.00		0.15	0.00		0.07	0.00		0.04	0.00	

TABELA 9 – Parâmetros de Avaliação no processo de Segmentação das imagens Padrões para o grupo de Crianças (10 volumes).

5.2 VE simulado: Imagens com Ruído

Nas Tabelas 10, 11 e 12 são apresentados os parâmetros de avaliação no processo de segmentação das imagens com Ruído Poisson.

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.15$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
HOMENS	1	92.71	4.55	4	93.76	4.20	4	92.88	4.12	5	92.79	4.51	5
	2	89.50	5.32	5	92.64	4.51	4	91.11	5.55	5	91.37	5.21	5
	3	92.14	4.32	4	89.77	5.62	5	92.31	5.01	4	91.78	4.55	4
	4	91.78	4.12	4	90.13	4.99	5	90.11	4.32	5	90.62	5.72	5
	5	93.57	4.88	3	89.30	5.05	5	91.78	4.88	4	90.05	4.98	5
	6	89.98	5.44	5	92.45	4.40	3	93.23	4.01	3	91.59	5.34	5
	7	90.45	5.25	5	89.70	5.34	5	91.55	4.93	5	91.00	4.48	4
	8	93.78	3.78	3	93.56	3.88	3	93.23	4.53	4	91.71	4.78	4
	9	92.23	4.98	4	91.32	5.26	5	91.45	4.69	5	90.79	5.02	5
	10	90.01	5.81	5	91.78	4.55	4	91.33	4.83	5	92.32	4.91	5
Media		91.62	4.85	5	91.44	4.78	5	91.90	4.69	5	91.40	4.95	5
Desv. Padrão		1.54	0.64		1.65	0.56		1.01	0.46		0.82	0.40	

TABELA 10 – Parâmetros de Avaliação no processo de Segmentação das imagens com Ruído Poisson para o grupo de Homens (10 volumes).

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.15$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
MULHERES	1	93.22	3.98	4	93.49	3.99	4	94.01	3.81	3	91.40	3.96	4
	2	91.78	4.88	4	93.78	3.78	3	91.43	4.59	4	93.20	4.23	3
	3	92.31	5.01	4	91.59	4.78	4	93.56	3.88	3	93.23	4.01	3
	4	93.23	4.01	3	91.80	4.54	4	92.31	5.01	4	92.86	5.78	4
	5	90.59	4.78	4	90.68	5.78	4	93.23	4.01	3	91.78	4.88	4
	6	90.68	5.78	4	91.78	4.88	4	90.68	5.78	4	92.53	4.52	4
	7	93.78	3.78	3	92.31	5.01	4	91.78	4.88	4	91.59	4.78	4
	8	91.87	4.50	4	93.56	3.88	3	91.59	4.78	4	93.91	5.01	4
	9	91.78	4.55	4	93.23	4.01	3	91.32	4.67	4	91.78	4.55	4
	10	92.56	3.88	3	92.23	4.98	4	93.78	3.78	3	91.78	4.88	4
Media		92.18	4.52	4	92.45	4.56	4	92.37	4.52	4	92.41	4.66	4
Desv. Padrão		1.06	0.63		1.03	0.64		1.18	0.65		0.86	0.54	

TABELA 11 – Parâmetros de Avaliação no processo de Segmentação das imagens com Ruído Poisson para o grupo de Mulheres (10 volumes).

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.15$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
CRIANÇAS	1	93.40	4.07	4	94.16	4.12	4	93.08	4.01	4	94.24	3.98	3
	2	93.23	4.01	3	93.78	3.78	3	91.78	4.88	4	93.50	3.78	3
	3	93.23	4.01	3	93.23	4.01	3	93.23	4.01	3	91.78	4.55	4
	4	90.68	5.78	4	91.78	4.88	4	91.78	5.01	4	92.68	5.78	4
	5	91.08	4.55	4	92.31	5.01	4	93.56	3.88	3	93.23	4.01	3
	6	92.23	4.01	3	91.59	4.78	4	90.68	5.78	4	92.02	5.01	4
	7	92.31	5.01	4	91.08	4.55	4	91.78	4.88	4	91.59	4.78	4
	8	91.78	4.88	4	91.78	4.55	4	91.59	4.78	4	91.75	5.78	4
	9	90.50	4.34	4	93.23	4.01	3	92.31	5.10	4	91.78	4.88	4
	10	93.56	3.88	3	91.78	4.55	4	93.78	3.78	3	91.78	4.55	4
Media		92.20	4.45	4	92.47	4.42	4	92.36	4.61	4	92.44	4.71	4
Desv. Padrão		1.16	0.61		1.05	0.42		1.01	0.66		0.93	0.70	

TABELA 12 – Parâmetros de Avaliação no processo de Segmentação das imagens com Ruído Poisson para o grupo de Crianças (10 volumes).

5.3 VE simulado: Imagens Filtradas

Nas Tabelas 13, 14 e 15 listam-se os parâmetros de avaliação no processo de segmentação após a etapa de filtragem, utilizando o filtro de Anscombe/Wiener. Como foi detalhado na secção 4.6, este filtro representa a abordagem mais apropriada para este propósito.

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.10$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
HOMENS	1	97.10	2.72	3	97.76	2.32	3	97.88	2.78	2	97.53	2.31	2
	2	98.42	2.11	2	97.88	2.31	3	98.02	2.43	2	97.93	2.22	2
	3	97.32	2.52	3	97.76	2.43	3	98.15	2.40	2	98.54	2.30	2
	4	97.76	2.43	3	98.01	2.20	3	98.65	1.92	1	97.66	2.21	2
	5	98.77	2.04	2	98.33	2.13	2	98.18	2.21	2	98.94	2.12	1
	6	98.13	2.34	3	98.23	2.01	2	97.99	2.20	2	98.84	1.94	1
	7	97.72	2.41	3	98.15	2.40	2	97.64	2.65	3	98.50	2.39	2
	8	98.56	2.02	2	97.95	2.36	3	98.83	1.98	1	98.97	1.90	1
	9	98.01	2.50	3	98.88	1.98	2	98.76	2.00	2	98.15	2.20	2
	10	98.88	2.00	2	97.88	2.43	3	98.81	1.76	1	98.97	1.92	1
Media		98.07	2.31	3	98.08	2.26	3	98.29	2.23	3	98.40	2.15	2
Desv. Padrão		0.60	0.25		0.34	0.17		0.43	0.33		0.55	0.18	

TABELA 13 – Parâmetros de Avaliação no processo de Segmentação após a etapa de filtragem através do Filtro Anscombe/Wiener para o grupo de Homens (10 volumes).

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.10$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}	%VP	%FP	Max_{dist}
MULHERES	1	98.50	2.86	2	98.15	2.40	2	98.65	2.12	1	98.02	2.32	2
	2	97.25	2.89	3	98.13	2.31	2	98.12	2.18	2	98.15	2.40	2
	3	97.70	2.42	3	97.76	2.43	2	98.23	2.30	2	98.01	2.34	2
	4	98.21	2.36	2	97.88	2.31	1	98.23	2.36	2	98.65	2.12	1
	5	98.04	2.92	3	98.13	2.14	2	98.67	1.86	1	98.76	1.98	1
	6	98.65	2.40	2	98.44	2.29	2	98.15	2.40	2	98.26	2.32	2
	7	98.15	2.40	2	97.88	2.31	2	98.45	2.01	1	98.67	1.87	1
	8	97.78	2.40	3	97.76	2.43	2	98.11	2.34	2	98.63	2.03	2
	9	97.88	2.31	3	98.47	1.98	1	98.65	2.12	1	98.44	2.21	2
	10	98.45	2.11	2	98.62	1.93	1	98.66	1.98	1	98.73	1.80	1
Media		98.06	2.51	3	98.12	2.25	2	98.39	2.17	2	98.43	2.14	2
Desv. Padrão		0.42	0.28		0.31	0.18		0.25	0.18		0.30	0.21	

TABELA 14 – Parâmetros de Avaliação no processo de Segmentação após a etapa de filtragem através do Filtro Anscombe/Wiener para o grupo de Mulheres (10 volumes).

Grupo	Teste	Algoritmo Genérico			Algoritmo LIFO			Algoritmo kFOEMS ($\theta=0.10$)			Algoritmo Pesos Dinâmicos		
		%VP	%FP	Max _{dist}	%VP	%FP	Max _{dist}	%VP	%FP	Max _{dist}	%VP	%FP	Max _{dist}
CRIANÇAS	1	97.83	2.84	2	97.79	2.34	2	98.65	2.12	1	98.42	2.21	2
	2	97.88	2.31	2	98.15	2.40	2	98.12	2.21	2	98.25	2.31	2
	3	98.10	2.30	2	98.74	2.00	1	98.13	2.34	2	98.74	1.92	1
	4	98.33	2.12	1	97.74	2.49	2	98.75	2.31	1	98.15	2.30	2
	5	98.15	2.40	2	97.88	2.31	2	98.45	2.12	1	98.65	2.12	1
	6	98.32	2.32	2	98.12	2.21	2	98.65	1.86	1	98.88	1.98	1
	7	97.72	2.44	2	98.35	2.11	1	98.20	2.17	2	98.50	2.42	2
	8	98.48	2.08	1	97.70	2.43	2	98.15	2.40	2	98.35	2.40	2
	9	97.87	2.32	2	98.65	2.12	1	98.64	2.31	2	98.15	2.34	2
	10	98.50	2.15	1	98.35	2.20	2	98.74	2.02	1	98.79	1.93	1
Media		98.12	2.33	2	98.15	2.26	2	98.45	2.19	2	98.49	2.19	2
Desv. Padrão		0.28	0.22		0.37	0.16		0.27	0.16		0.27	0.19	

TABELA 15 – Parâmetros de Avaliação no processo de Segmentação após a etapa de filtragem através do Filtro Anscombe/Wiener para o grupo de Crianças (10 volumes).

5.4 Phantom do VE gerado por meio do NCAT-4D

A Tabela 16 mostra os valores dos parâmetros de avaliação (VP, FP e Max_{dist}) após execução dos quatro algoritmos de segmentação no *phantom* gerado por meio do NCAT-4D. Na etapa de filtragem, sendo conhecidas as vantagens frente às outras abordagens implementadas, foi utilizado o filtro de Anscombe/Wiener. Pode-se revisar alguns parâmetros importantes deste *phantom* no Anexo B.

Algoritmo	Parâmetros de Avaliação	PHANTOMS NCAT-4D		
		Imagens Padrões	Imagens com Ruído	Imagens Filtradas
Genérico	%VP	99.24 ± 0,05	89.55 ± 6.75	92.12 ± 1.48
	%FP	0.42 ± 0.02	6.75 ± 1.84	2.51 ± 0.98
	MaxDist	1 pixel	5 pixels	2 pixels
LIFO	%VP	99.38 ± 0,04	89.95 ± 6.17	92.58 ± 1.05
	%FP	0.45 ± 0.04	6.15 ± 1.12	2.11 ± 0.88
	MaxDist	1 pixel	5 pixels	2 pixels
KFOEMS- θ ($\theta=0.15$)	%VP	99.40 ± 0,05	90.07 ± 6.27	93.09 ± 1.81
	%FP	0.49 ± 0.05	6.05 ± 1.48	2.23 ± 0.84
	MaxDist	1 pixel	5 pixels	2 pixels
Pesos Dinâmicos	%VP	99.64 ± 0,03	91.51 ± 5.57	94.05 ± 1.32
	%FP	0.27 ± 0.03	5.34 ± 1.23	2.31 ± 0.93
	MaxDist	1 pixel	5 pixels	2 pixels

TABELA 16 – Parâmetros de Avaliação no processo de Segmentação das imagens Padrões, com Ruído e Filtradas (usando o Filtro de Anscombe/Wiener) do *Phantom* ncat_LV60bpm_log que representa o VE do coração.

5.5 Imagens PET reais

Nas Tabelas 17, 18, 19 e 20 são apresentados os resultados de quantificação dos quatro volumes PET reais analisados. Como foi indicado na seção 4.7, note-se que temos considerado unicamente os testes de filtragem que representaram os melhores resultados, junto com cada um dos algoritmos de segmentação implementados neste projeto.

Os valores de quantificação estão sendo representados em duas unidades diferentes de volume: *voxels* e mm^3 .

Volume	Segmentador	Filtro da Mediana (Janela 5x5)	Filtro da Gauss. Ponderada (Sigma 1.5)	Filtro de Anscombe/Lee
Volume N° 33 Código - 2130000	Genérico	16226 voxels	16393 voxels	18234 voxels
		(81459 mm3)	(82294 mm3)	(91538 mm3)
	LIFO	17060 voxels	17199 voxels	18830 voxels
		(85643 mm3)	(86339 mm3)	(94529 mm3)
	KTetaFOEMS (0.10)	19934 voxels	19500 voxels	23328 voxels
		(100071 mm3)	(97890 mm3)	(117107 mm3)
	Pesos Dinâmicos	24154 voxels	24078 voxels	24929 voxels
		(121254 mm3)	(120872 mm3)	(125146 mm3)

TABELA 17 – Resultados da quantificação do volume PET N° 33.

Volume	Segmentador	Filtro da Mediana (Janela 5x5)	Filtro da Gauss. Ponderada (Sigma 1.5)	Filtro de Anscombe/Lee
Volume N° 34 Código - 2430000	Genérico	16675 voxels	16912 voxels	18873 voxels
		(83711 mm3)	(84903 mm3)	(94746 mm3)
	LIFO	17997 voxels	18317 voxels	19476 voxels
		(90347 mm3)	(91954 mm3)	(97774 mm3)
	KTetaFOEMS (0.10)	19432 voxels	19597 voxels	20546 voxels
		(97553 mm3)	(98379 mm3)	(103144 mm3)
	Pesos Dinâmicos	22162 voxels	22173 voxels	22960 voxels
		(111256 mm3)	(111309 mm3)	(115263 mm3)

TABELA 18 – Resultados da quantificação do volume PET N° 34.

Volume	Segmentador	Filtro da Mediana (Janela 5x5)	Filtro da Gauss. Ponderada (Sigma 1.5)	Filtro de Anscombe/Lee
Volume N° 35 Código - 2730000	Genérico	18281 voxels	18904 voxels	19746 voxels
		(91773 mm3)	(94901 mm3)	(99128 mm3)
	LIFO	19593 voxels	20403 voxels	21420 voxels
		(98357 mm3)	(102425 mm3)	(107533 mm3)
	KTetaFOEMS (0.10)	23326 voxels	23374 voxels	24707 voxels
		(117101 mm3)	(117338 mm3)	(124032 mm3)
	Pesos Dinâmicos	26064 voxels	25956 voxels	27142 voxels
		(130842 mm3)	(130302 mm3)	(136257 mm3)

TABELA 19 – Resultados da quantificação do volume PET N° 35.

Volume	Segmentador	Filtro da Mediana (Janela 5x5)	Filtro da Gauss. Ponderada (Sigma 1.5)	Filtro de Anscombe/Lee
Volume N° 36 Código - 3030000	Genérico	16550 voxels	17347 voxels	92305 voxels
		(83081 mm3)	(80046 mm3)	(92305 mm3)
	LIFO	18817 voxels	17597 voxels	19172 voxels
		(94463 mm3)	(88339 mm3)	(96245 mm3)
	KTetaFOEMS (0.10)	20070 voxels	21435 voxels	23805 voxels
		(100751 mm3)	(107603 mm3)	(119501 mm3)
	Pesos Dinâmicos	24787 voxels	24622 voxels	25642 voxels
		(124433 mm3)	(123604 mm3)	(128597 mm3)

TABELA 20 – Resultados da quantificação do volume PET N° 36.

Capítulo 6

DISCUSSÃO

6.1 Importância e contribuições da pesquisa

Temos vários fatos pelos quais nosso projeto tem uma significação relevante no processamento de Imagens de Medicina Nuclear na modalidade PET. Estas contribuições serão apresentadas a seguir:

1. Comprovou-se que os algoritmos baseados na teoria Fuzzy Connectedness são bastante adequados para a Segmentação de imagens 3D, e facilmente escaláveis para n-dimensões. Além disso, os algoritmos utilizados apresentaram robustez e bons resultados quando testados nos diferentes volumes simulados, *phantoms* e volumes reais.
2. A segmentação das imagens sem ruído (padrões) tem o intuito de indicar se a implementação dos métodos de segmentação estão corretos. Nesse sentido, quando se realizou este processo com as imagens do VE simulado, foi possível obter praticamente 100% de VP e 0% de FP nos quatro algoritmos de segmentação utilizados.
3. Embora o Ventrículo Esquerdo simulado seja muito simples, permitiu fazer uma análise completa com diversos volumes (30 volumes diferentes) e deu a possibilidade de testar nossa estrutura de processamento, bem como analisar os parâmetros que foram encontrados.
4. A utilização de *phantoms* gerados através de softwares conhecidos no ambiente médico, como é o caso do NCAT-4D, foi realmente de muita utilidade, já que foi possível não só testarmos os algoritmos

de filtragem e os algoritmos segmentadores, mas também comprovamos a relevância que tem este tipo de ferramentas no processo de representação de imagens médicas simuladas, permitindo obter imagens muito próximas a aquelas obtidas nos exames médicos reais.

5. Vários trabalhos já mostraram a efetividade do filtro de Anscombe/Wiener para imagens de Medicina Nuclear. O presente trabalho mostrou, dentro das hipóteses assumidas e dos filtros considerados, que o filtro de Anscombe/Wiener também proporciona os melhores resultados, considerando como critério funcional a qualidade da segmentação. Este ponto é importante e tem impacto na quantificação em imagens de Medicina Nuclear.

Capítulo 7

CONCLUSÕES

Baseados nos resultados do teste de avaliação, feito sob as imagens padrões (sem ruído), pode-se certificar que os quatro algoritmos de segmentação estão bem implementados (valores aproximados de 100% para VP e de 0% para FP).

Ficou evidenciado, por meio do teste estatístico Qui-Quadrado (Chi-Square) que o ruído introduzido nos volumes do VE simulado através do código de MatLab corresponde, sim, ao ruído Poisson, já que as intensidades destes volumes simulados estão dentro das intensidades que foram examinadas e cujas hipóteses não foram rejeitadas. Por outro lado, no caso do *phantom* do VE gerado por meio do NCAT-4D, a hipótese de que o ruído seja Poisson, foi rejeitada. Mesmo assim, pode-se afirmar que os procedimentos de filtragem utilizados neste projeto, permitiram melhorar a qualidade das imagens independente na natureza do ruído (Poisson ou Gaussiano).

O presente trabalho permitiu demonstrar e corroborar a supremacia que o Filtro de Anscombe/Wiener apresentou em comparação aos outros filtros testados, tendo em consideração as características dos volumes PET analisados e os objetivos e/ou propósitos específicos deste projeto.

Dentro dos quatro algoritmos de segmentação utilizados, o algoritmo FC que utiliza Pesos Dinâmicos apresentou os melhores resultados. É necessário indicar também que esta abordagem mesmo que tenha apresentado um ótimo desempenho como segmentador, o tempo de processamento utilizado foi maior, comparado com os tempos dos outros algoritmos (Ver Anexo G). Esta diferença do tempo computacional

ocorre, pois o Algoritmo baseado em Pesos Dinâmicos, faz um processo interno para finalmente obter os pesos adequados para realizar esta tarefa.

Foram determinados, como produto da análise ROC, que os valores para $w_1 = 0.3$ e $w_2 = 0.7$ representam os pesos que permitem obter os melhores resultados de segmentação usando os algoritmos Genérico, LIFO e KTetaFOEMS.

Os resultados encontrados, com a utilização dessa metodologia, mostraram-se satisfatórios. Foi obtido, para o caso do VE simulado, 2.19 % de Falsos Positivos e 98.49 % de Verdadeiros Positivos no final da sequência de processamento instaurada, assim como 2.31 % de Falsos Positivos e 94.05 % de Verdadeiros Positivos para o *phantom* gerado por meio do NCAT-4D, utilizando para ambos casos na etapa de filtragem o Filtro de Anscombe/Wiener e na etapa de segmentação o algoritmo baseado em Pesos Dinâmicos.

Pelo fato de não dispor das imagens gold standard dos volumes PET reais, não foi possível submeter estes volumes na estrutura de processamento implementada no projeto. No entanto, usando as escolhas feitas ao longo do projeto por meio dos testes tanto dos volumes do VE simulado quanto do *phantom* gerado através do NCAT-4D, foi possível quantificar um conjunto de volumes PET reais.

Capítulo 8

DESAFIOS E PROJETO FUTURO

O próximo passo será o desenvolvimento do projeto de doutorado, o qual representa para nós um desafio muito alto, já que reúne um conjunto de assuntos de atualidade e uma aplicabilidade altamente clínica que, segundo os especialistas, teria um grande impacto como ferramenta de apoio na área médica.

O projeto de doutorado tem como título: “Análise da Dinâmica e Quantificação Metabólica de imagens de Medicina Nuclear na modalidade PET/CT”. Neste projeto pretende-se investigar a dinâmica e a quantificação metabólica de imagens médicas 3D dinâmicas da modalidade PET/CT. Entretanto, sabe-se que a quantificação em Medicina Nuclear é dificultada por diversos fatores, entre os quais estão a correção de atenuação, espalhamento, algoritmos de reconstrução e modelos assumidos. Neste contexto, o principal objetivo deste projeto é melhorar a precisão na análise de imagens de PET/CT via processos realísticos e bem controlados.

Para esse fim, estamos propondo a elaboração de uma estrutura modular, a qual estará conformada por um conjunto de passos consecutivamente interligados, representados no Diagrama de Blocos da Figura 27, que se iniciam na simulação de *phantoms* antropomórficos 3D dinâmicos, para posteriormente gerar as projeções realísticas PET/CT usando a plataforma GATE (com simulação de Monte Carlo). Em seguida, será aplicada uma etapa de pré-processamento, mediante o filtro de Anscombe/Wiener para a redução de ruído Poisson próprio deste tipo de imagens. Tendo projeções de melhor qualidade após a filtragem, uma etapa de reconstrução de imagens 3D dinâmicas será aplicada, e

consequentemente as imagens serão filtradas e segmentadas (baseados na teoria Fuzzy Connectedness). Uma vez definida a região de interesse (ROI), serão produzidas suas respectivas Curvas de Atividade de Entrada e Resultante requeridas no processo de análise da dinâmica de compartimentos, com o qual será obtida a Quantificação do Metabolismo do órgão ou estrutura de estudo. Finalmente, procedimentos de avaliação serão utilizados como métrica para conhecer o nível de eficiência e precisão da aplicação. Analogamente, para certificar a validade do processo, serão analisadas imagens reais fornecidas pelo Centro de Medicina Nuclear do HC e pelo InCor - HCFMUSP.

Figura 27 – Diagrama de Blocos do Projeto de Doutorado

REFERÊNCIAS

Abe R. Otimização dos Parâmetros do Filtro Butterworth na reconstrução tomográfica em SPECT Cardíaco. Tese de mestrado, Faculdade de Engenharia Elétrica e de Computação da UNICAMP. 1999.

Adams E., Asua J., Conde Olasagasti J., Erlichman M., Flynn K., Hurtado-Saracho I.. Positron Emission Tomography: Experience with PET and Synthesis of the Evidence. A Joint Project Produced on Behalf of: International Network of Agencies for Health Technology Assessment (INAHTA). 1999.

Anscombe F. J. The transformation of Poisson, Binomial and Negative Binomial Dat. *Biometrika* 1948; 15:246-254.

Araújo E. B.; Lavinias T.; Colturato M. T.; Mengatti J. Garantia da qualidade aplicada à produção de radiofármacos. *Revista Brasileira de Ciências Farmacêuticas*. vol.44 no.1 São Paulo Jan./Mar. 2008.

Bethea R.M.; Duran R.A. Statistical methods for engineers and scientists. Mc Hill Reckker, New York, NY, 1995.

Blue Cross and Blue Shield Association. FDG-Positron mission tomography for Evaluating Breast Cancer. Assessment Program, v. 16, n. 5, 74 p., 2001.

Bushberg J.T., Seibert J.A., Leidholdt E.M., Boone J.M. The Essencial Physics of Medical Imaging. Lippincott Williams & Wilkins Edition. Chapter 1: Introduction to Medical Imaging. Pages 3-17, Second Edition, 2002.

Braga A. C. S. Curvas ROC: Aspectos funcionais e aplicações. Tese de Doutorado, Universidade do Minho, 2003.

Bruce, R. Hybrid Systems, PET dominates nuclear medicine. *Diagnostic Imaging On-Line*, November 22, 2000.

Budinger, T.F. PET Instrumentation: What are the Limits? *Seminars in Nuclear Medicine*, v. XXVIII, n. 3, p. 247-267, 1998.

Caetano, R. Inovações e trajetórias tecnológicas no território das imagens médicas. Tese de Doutorado em Saúde Coletiva. Instituto de Medicina Social, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. 2002.

Carvalho B.M., Gau C.J., Herman G.T., Kong T.Y.. Algorithms for fuzzy segmentation. *Pattern Analysis Application*, 2(1):73–81, April 1999.

De Almeida J. P., Trindade M. V., Gomes D., Fernandes C.D., de Moraes W. O que é Medicina Nuclear? Portal dos profissionais de Radiologia. <http://www.rxinfo.com.br/>, 2002.

Dial R. Algorithm 360: Shortest path forest with topological ordering. *Commun ACM* 12, 632-633, 1969.

De Amorin D.P.M.L. Modelos Fenomenológicos de Distribuição Intercompartimental de Substâncias Radioactivas. Faculdade de Engenharia da Universidade de Porto. 2007.

Dijkstra E. A note on two problems in connection with graphs. *Numer math* 1, 269-271, 1959.

Fabey, F.H. State of the art in emission tomography equipment. *Radiographics*, v. 16, n. 2, p. 409-20, 1996.

Falcão, A. X.; Stolfi, J.; Lotufo, R. A. The image foresting transform: theory, algorithms, and applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* Volume 26, Issue 1, Page(s):19 – 29, 2004.

Fifer, A.M.; Grossman W. Measurement of Ventricular Volumes, Ejection Fraction, Mass, Wall Stress, and Regional Wall Motion. Section V: Evaluation of Cardiac Function. Harvard Medical School; Coronary Care Unit, Massachusetts General Hospital, Boston, 2000.

Furuie S.S.; Jara G.; Rebelo M.F.S.; Melo C.P. Segmentação do Ventriculo Esquerdo Cardiaco Usando Modelo de Expansão de Balao. *Revista Brasileira de Engenharia Biomédica*, v. 12, n. 3, p. 99-108, 1996

Gonzalez, R. C.; Woods, R. E. *Digital Image Processing*. 3rd ed, New Jersey, Pearson Prentice Hall, 2008. Chap.9 – Morphological Image Processing.

Homen M. R. P., Mascarenhas N. D. A., Costa L. F., Preza C. Biological Image Restoration in Optical – Sectioning Microscopy Using Prototype Image Constraints. *Real-Time Imaging* 8, 475-490. 2002.

Homen M. R. P., Mascarenhas N. D. A., Costa L. F. Linear Filters for Deconvolution Microscopy. 6-th IEEE Southwest Symposium on Image Analysis and Interpretation Proceedings, Lake Tahoe. v.1 p. 142-146. 2004.

Inouye T. Square Root Transform for the Analysis of Quantum Fluctuations in Spectrum Data. *Nuclear Instruments and Methods*, 1971.

Instituto Oncoguia, 2009. <http://www.oncoguia.com.br/site/index.php/>.

Johnson S., Bigelow C., Lee K., Pryor D., Jones R., "Prediction of death and myocardial infarction by radionuclide angiocardiology in patients with suspected coronary artery disease," *American Journal of Cardiology*, vol. 67, pp. 919-926, 1991.

Jones DN, McCowage GB, Sostman HD, Brizel DM, Layfield L, Charles HC, Dewhirst MW, Prescott DM, Friedman HS, Harrelson JM, Scully SP, Coleman RE. Monitoring of neoadjuvant therapy response of soft-tissue and musculoskeletal sarcoma using fluorine-18-FDG-PET. *J Nucl Med* 1996.

Knapp Jr., FF.; Mirzadeh, S. The continuing important role of radionuclide generator systems for nuclear medicine. *European Journal of Nuclear Medicine*, v. 20, n. 12, p. 1151-1165, 1994.

Kass M., Witkin A., Terzopoulos D., "Snakes: Active Contour Models," *International Journal of Computer Vision*, pp. 321-331, 1988.

Lage D. M. Visualização de artérias coronárias epicárdicas em imagens ecocardiográficas tridimensionais com contraste de microbolhas. Tese de doutorado, Faculdade de Medicina da Universidade de São Paulo. Programa de Cardiologia. 2010.

Lee J. S. Digital image enhancement and noise filtering by use of local statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, pp. 165-168, 1980. (Adaptive Wiener filter)

Leitão G.; Mariz D. Probabilidade e Estatística. Planejamento de Capacidade de Sistemas. Instituto Federal de Educação, Ciência e Tecnologia. Rio Grande do Norte, 2010.

López J. Técnicas avanzadas de reconstrucción de imagen nuclear PET, X-CT y SPECT. Memoria del Trabajo del Máster de Física Biomédica. Madrid. 2008.

Mascarenhas N.D.A., Santos C.A.N., Cruvinel P.E. Transmission Tomography under Poisson noise using the Anscombe Transform and a Wiener Filter of the projections. *Nuclear Instruments and Methods in Physics Research A*. v. 423, p. 265-271. 1999.

Margotto P. R. Curva ROC, como fazer e interpretar no SPSS. Escola Superior de Ciências da Saúde, 2010.

Marques Filho O.; Vieira N.H. Processamento Digital de Imagens. Editora Brasport, Rio de Janeiro, 1999. ISBN 8574520098.

Matej S, Surti S, Jayanthi S, Daube-Witherspoon ME, Lewitt RM, Karp JS. Efficient 3D TOF PET Reconstruction Using View-Grouped Histo-Images: DIRECT – Direct Image Reconstruction for TOF. *IEEE Transactions on Medical Imaging*, 2008.

Moraes M. C, Furuie S. S. Automatic coronary wall segmentation in intravascular ultrasound images using binary morphological reconstruction. *Ultrasound in Medicine & Biology* 2011; 37(9):1486-1499.

National Academy of Sciences, U.S.. Mathematics and Physics of Emerging Biomedical Imaging. National Academy of Engineering/ Institute of Medicine. Washington DC: National Academy Press, 233p., 1996.

Nist/Sematech. Handbook of Statistical Methods, access link: <http://www.itl.nist.gov/div898/handbook/>, 2011.

Nyúl, A. L. G; Falcão, A. X; Udupa, J. K. “Fuzzy-connected 3D image segmentation at interactive speeds”. *Elsevier on Graphical Models*, v.64, p. 259-281, 2002.

Oliveira R., Santos D., Ferreira D., Coelho P., Veiga F. Preparações radiofarmacêuticas e suas aplicações. *Brazilian Journal of Pharmaceutical Sciences*, vol. 2, n. 2, jun 2006.

Osher, S; Fedkiw, R. Level Set Methods and Dynamic Implicit Surfaces. Springer-Verlag, New York, 2002.

Otsu N. Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man and Cybernet*, 9, N° 1, pp. 62-66. 1979.

Pednekar, A. S.; Kakadiaris, I. A. Image Segmentation Based on Fuzzy Connectedness using Dynamic Weights, 2006.

Pedone M.D.; Castro I.; Hatem D.; Haertel J.C.; Feier F.; Pandolfo F. Variações de Parâmetros da Função Diastólica do Ventrículo Esquerdo de acordo com a Idade através da Ecocardiografia com Doppler Tissular. *Sociedade Brasileira de Cardiologia*, vol. 83, N° 6, 2004.

Pertusa G.J.F. Técnicas de Análisis de Imágenes: Aplicaciones en Biología. Editorial Maite Simon, Grupo Carduche. Capítulo 9. Procesado de Imagen, pag. 141-161, 2003.

Peregrino A.A.F.; Coelho C.D.; Vianna C.M.M.; Coeli C.M.; Vianna D.; Werneck G.L.; Magalhães R.E.A.; Caetano R. Síntese das Avaliações Tecnológicas sobre PET realizadas por Agências Internacionais de ATS na área de Oncologia. Avaliação Tecnológica da Tomografia de Emissão de Pósitrons (PET): Revisão Sistemática e Proposta de Estudo. 2004.

Pozzo L. Tomografia por emissão de pósitrons com sistemas PET/SPECT: Um estudo da viabilidade de Quantificação. Tese de Doutorado do Instituto de Física da Universidade de São Paulo. 2005.

Robert, G.; Milne, R. Positron emission tomography: establishing priorities for health technology assessment. *Health Technology Assessment*, v. 3, n. 16, p. 1-54, 1999.

Robilotta C.C. A tomografia por emissão de pósitrons: uma nova modalidade na medicina nuclear brasileira. *Rev Panam Salud Publica*. 2006; 20(2/3):134–42.

Rodrigues I., Sanches J., Bioucas-Dias J.. Denoising of medical images Corrupted by poisson noise. *Image Processing*, 2008. ICIP 2008. 15th IEEE International Conference, páginas 1756–1759, 2008.

Romualdo L.C.S. Restauração de imagens mamográficas digitais utilizando o filtro de Wiener no domínio de Anscombe e o filtro inverso da MTF no domínio da frequência. Tese de Mestrado, Escola de Engenharia de São Carlos da Universidade de São Paulo, 2009.

Russ J. C. *The Image Processing Handbook*. 3rd ed, North Carolina, a CRC Handbook published in cooperation with IEEE press. 1998, Chap.9 – 3D Image Acquisition.

Sabbatini R., *The PET Scan: A New Window Into the Brain*. Brain & Mind, Electronic Magazine on Neuroscience, 1997.

Sarmiento V.A.; Carvalho A.C.R; Luz I.M. Evaluation of photon noise on digitalized images. *Revista da Faculdade de Odontologia, Porto Alegre*, vol. 45, N°1, pg. 23-28, 2004.

Sergars W.P.; Tsui B.M.W. MCAT TO XCAT: The Evolution of 4-D Computerized Phantoms for Imaging Research. *IEEE*, Vol. 97, No. 12, 2009.

Shaw L., Heinle S., Borges-Neto S., Kesler K., Coleman R., Jones R., "Prognosis by measurements of left ventricular function during exercise," *Journal of Nuclear Medicine*, vol. 39, pp. 140-146, 1998.

Shaw I. S., Simões M. G. *Controle e Modelagem Fuzzy*. Edgar Blücher e FAPESP. 1999.

Souza S.; De Oliveira A. Alguns comentários sobre a Teoria Fuzzy. *Exacta versão 1. Matematica Aplicada IME – USP*. 1992. p. 139 – 147.

Veress A. I., Segars W. P., Weiss J. A., Tsui B. M. W., Gullberg G. T.. Normal and Pathological NCAT Image and Phantom Data Based on Physiologically Realistic Left Ventricle Finite Element Models. *IEEE Transactions on Medical Imaging*. 2006.

Tinois E., *Imagem Funcional – PET e fMRI*. MultiCiência: Tecnologia para a Saúde #5, 2005.

Udupa J.K., Samarasekera S., “Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation”, *Graphical Models Image Process*. 58 (3) (1996) 246–61.

Udupa KJ, LaBlanc RV, Schmidt H. A Methodology for Evaluating Image Segmentation Algorithms 2002.

Udupa, J. K.; Saha, P. K. Fuzzy Connectedness and Image Segmentation. IEEE, Proceedings of IEEE, Vol. 91, pp. 1649-1669, October 2003.

Wagner Jr. H.N. A brief history of positron emission tomography (PET). Seminars in Nuclear Medicine, v. 28, n°. 3, p. 213-20, 1998.

Whitehouse D.J. Handbook of Surface Metrology. Leicester - England, 1998.

Wikipedia. http://pt.wikipedia.org/wiki/Ventr%C3%ADculo_card%C3%ADaco, 2011.

Xu, J. P. Generalized Gradient Vector Flow External Forces for Active Contours. Signal Processing 71, pp 131-139, 1998.

ANEXO A

PARÂMETROS DE AQUISIÇÃO DO VE SIMULADO

Primeiro grupo: Homens adultos										
Parâmetro	Caso_H01	Caso_H02	Caso_H03	Caso_H04	Caso_H05	Caso_H06	Caso_H07	Caso_H08	Caso_H09	Caso_H10
Distância máxima em x	30	30	28	29	29	29	31	31	32	32
Distância máxima em y	30	28	28	29	29	31	32	30	31	32
Distância máxima em z	40	39	41	38	38	39	42	40	38	41
Espessura da parede	8	9	10	11	12	8	9	10	11	12
Intensidade do background	2	2	2	2	2	3	3	3	4	4
Intensidade do objeto	20	20	20	20	20	19	21	22	21	22

Segundo grupo: Mulheres adultas										
Parâmetro	Caso_M01	Caso_M02	Caso_M03	Caso_M04	Caso_M05	Caso_M06	Caso_M07	Caso_M08	Caso_M09	Caso_M10
Distância máxima em x	20	20	18	18	19	19	21	21	22	22
Distância máxima em y	20	19	21	22	19	18	22	21	20	18
Distância máxima em z	30	32	31	28	29	32	30	29	28	31
Espessura da parede	8	9	10	11	12	8	9	10	11	12
Intensidade do background	2	2	2	2	3	3	4	4	5	5
Intensidade do objeto	20	20	20	20	19	21	19	21	19	21

Terceiro grupo: Crianças										
Parâmetro	Caso_C01	Caso_C02	Caso_C03	Caso_C04	Caso_C05	Caso_C06	Caso_C07	Caso_C08	Caso_C09	Caso_C10
Distância máxima em x	13	13	11	11	12	12	14	14	15	15
Distância máxima em y	13	12	15	15	14	13	12	11	14	11
Distância máxima em z	18	20	16	19	17	20	18	16	19	17
Espessura da parede	8	9	10	8	9	10	8	9	10	8
Intensidade do background	2	2	2	2	3	3	4	4	5	5
Intensidade do objeto	20	20	20	20	10	19	21	22	23	24

Tabela 21 – Parâmetros dos volumes do VE utilizados na sequência de processamento, foram divididos em três grupos (Homens, Mulheres e Crianças) com dez diferentes volumes cada.

ANEXO B

PARÂMETROS DO *PHANTOM* DO VE ATRAVÉS DO NCAT-4D

Neste anexo, são mostrados os principais parâmetros adotados na reprodução do *phantom* utilizado no projeto.

```

ncat_LV60bpm_log

*** DYNAMIC NCAT PHANTOM, version 2.0 in C***
-----
PARAMETERS: (dynamic XCAT version 2.0 in C)
-----
act_phan_each = 1 (activity phantom each frame saved)
act_phan_ave = 0 (time-average activity phantom NOT saved)
atten_phan_each = 0 (atten. coeff. phantom frames NOT saved)
atten_phan_ave = 0 (time-average atten. coeff. phantom NOT saved)
Beating heart motion included only
Total Output Period = 1.0000 secs
Total Output Frames = 16
==>Time Per Frame = 0.0625 secs

beating heart period = 1.0000 secs
Heart Start Phase Index = 0.0000
Respiratory period = 5.0000 secs
Resp. Start Phase Index = 0.0000
maximum diaphragm motion = 2.0000
heart scale = 1.0000
hrt_lv_length = 9.4300
hrt_lv_radius = 2.9700
body long axis = 34.5000
body short axis = 27.5000
body height = 192.0000
rib long axis = 30.0000
rib short axis = 22.7000
air_flag = 0
breast_type = 1 (prone position model)
which_breast = 0 (no breasts, male)
breast_long_axis = 18.2000
breast_short_axis = 7.0000
breast_height = 14.0000
rdiaph_lv_position = 3.4000
ldiaph_position = 1.9000
intv (cm) = 1.0000
pixel width = 0.6250 (cm/pixel)
slice width = 0.6250 (cm/pixel)
voxel volume (pixel width^2)(slice_width) = 0.244141 ml
array_size = 64
subvoxel_index = 2 => 8 subvoxels per voxel
starting slice number = 228
ending slice number = 247
apical_thin = 1 (Apical thinning present)

Main orientation of heart:
zy_rotation (beta)= 0.00 (deg)
xz_rotation ( phi)= 90.00 (deg)
yx_rotation ( psi)= 0.00 (deg)

valve thickness <= 0 (no valves )
av_step = 0.3000 (smoothing between atria & ventr.)
total_rotation = 0 (no heart rotation during contraction )
activity_unit = 0 (no scaling of activity values)

radionuclide energy (keV)= 140.0000
-----

```

```

ncat_LV60bpm_log
-----
CREATING FRAME #1 ...

Time = 0.000 seconds
Activity Ratios
-----
Body = 1.3500
Liver = 0.0000
Gall Bladder = 0.0000
Lung = 0.0000
Stomach Wall = 0.0000
Stomach Contents = 0.0000
Kidney = 0.0000
Spleen = 0.0000
Vertebral Bone(head) = 0.0000
Vertebral Bone(process) = 0.0000
Pelvis Bone = 0.0000
Wet Rib Bone = 0.0000
Bone Cartilage = 0.0000
Abdom. Artery = 0.0000
Abdom. Vein = 0.0000
Bladder = 0.0000
Prostate = 0.0000
Testes = 0.0000
Ureter = 0.0000
Urethra = 0.0000
Lymph (normal) = 0.0000
Lymph (abnormal) = 0.0000

LV Myocardium = 100.0000
LA Myocardium = 0.0000
RV Myocardium = 0.0000
RA Myocardium = 0.0000

LV Blood pool = 2.7000
LA Blood pool = 0.0000
RV Blood pool = 0.0000
RA Blood pool = 0.0000
-----
Volume of LV myocardium = 175.537109 ml
Volume of RV myocardium = 44.952393 ml
Volume of LA myocardium = 18.920898 ml
Volume of RA myocardium = 13.092041 ml
Volume of LV chamber = 119.750977 ml
Volume of RV chamber = 172.027588 ml
Volume of LA chamber = 38.452148 ml
Volume of RA chamber = 50.476074 ml
-----
Volume of TOTAL HEART = 633.209229 ml

Left Ventricle Center of Mass (pixels):
X = 35.251739
Y = 29.317629
Z = 8.918637

Maximum activity per voxel = 100.000000
Total activity = 805327.437500

Saving activity phantom in file : imagens\lv60bpm\ncat_LV60bpm_act_1
-----

```

FIGURA 28 – Principais parâmetros do *phantom* ncat_LV60bpm_log. Arquivo fornecido pelo Serviço de Informática do InCor - HCFMUSP.

ANEXO C

DADOS EXAME PET REAL

Da base de dados do InCor-HCFMUSP utilizamos um exame PET dinâmico cardíaco de um arquivo chamado DICOM-xxxxx.7z, pertencente a um paciente de sexo masculino. Para visualizar a informação contida, foi preciso descompactar o arquivo por meio do programa 7-zip.

Dentro da estrutura do arquivo DICOM-xxxxx, encontramos a arvore com a distribuição seguinte:

Os dois últimos arquivos contem dados relevantes do exame PET, mostrados na Tabela 22, onde pode-se conhecer a quantidade total de volumes gerados no exame, os tempos de aquisição de cada um dos volumes e o consumo de radiofármaco em cada um destes períodos. Para propósito de nosso estudo, foi importante saber que foram gerados 36 volumes em total, já que o arquivo DINAMICOBARIOSEM continha um total de 1260 images com extensão DICOM, com o qual foi possível determinar que cada volume era constituído por 35 imagens.

Volume Number	Acquisition times (in seconds)		Radiopharm. Consumption
	Start	End	BG [mmol/l]
1	0	300	6.55
2	300	600	6.88
3	600	900	6.61
4	900	1200	6.44
5	1200	1500	6.49
6	1500	1800	6.22
7	1800	2100	6.66
8	2100	2400	5.72
9	2400	2700	5.55
10	2700	3000	5.83
11	3000	3300	4.66
12	3300	3600	4.94
13	3600	3900	4.77
14	3900	4200	4.88
15	4200	4500	4.55
16	4500	4800	4.55
17	4800	5100	4.72
18	5100	5400	4.94
19	5400	5700	5.16
20	5700	6000	4.44
21	6000	6300	4.88
22	6300	6600	5.00
23	6600	6900	5.00
24	6900	7200	5.16
25	7200	7500	4.94
26	7500	7800	4.72
27	7800	8100	4.88
28	8100	8400	4.88
29	8400	8700	5.00
30	8700	9000	5.05
31	9000	9300	4.88
32	9300	9600	4.83
33	9600	9900	4.05
34	9900	10200	4.72
35	10200	10500	4.94
36	10500	10800	4.94

TABELA 22 – Reporte do exame PET: Observam-se os tempos de aquisição dos 36 volumes PET, assim como o consumo do radiofármaco em cada período de tempo.

A codificação das imagens não ajuda no processo de seleção do grupo de imagens para a reconstrução de cada volume. Nesse sentido, depois de varios testes conseguimos identificar a forma certa de agrupação para cada volume. Como exemplo, apresentaremos o último volume (volume N° 36) para mostrar a codificação das imagens que a compoem, assim como alguns dados adicionais que serviram no processo de quantificação dos volumes.

Usando qualquer algoritmo de reconstrução 3D é possível carregar as 35 imagens indicadas e obter o volume PET requerido, como é mostrado na Figura 29:

FIGURA 29 – Volume PET N° 36 usando ferramentas de reconstrução 3D do ImageJ.

Sendo imagens DICOM, é possível conhecer a dimensão exacta de cada elemento (voxel) das imagens 3D com as informações obtidas no processo de reconstrução como mostra a Figura 29. Neste exame, temos imagens 220x220 mm que equivalem a 128x128 pelo qual cada pixel é aproximadamente igual a 2.95 mm^2 . Além disso, o espaçamento entre cada uma das 35 imagens que conformam os volumes é de 4.25 mm. Juntando estes dois valores, obtemos que cada voxel equivale a 12.55 mm^3 .

ANEXO D

GOODNESS OF FIT TEST

IMAGE 3D (Lambda = 5)									
k	Value (i)	Histogram	O _i	Media	P _{Lambdas(i)} Lambda = 5	F(Y)	E _i	Chi-square (X²)	Lambda = 5
1	0	14026	14026	0	0.006738	0.006738	14130.5465	0.773500	
2	1	70630	70630	70630	0.033690	0.033690	70652.7327	0.007314	Media = 5.0006
3	2	176616	176616	353232	0.084225	0.084225	176631.8319	0.001419	Variancia = 5.0074
4	3	294968	294968	884904	0.140374	0.140374	294386.3855	1.149083	Desvio_Padrão = 2.23773
5	4	367486	367486	1469944	0.175468	0.175468	367982.9831	0.671205	Dataplot = 0.67 ->
6	5	367616	367616	1838080	0.175468	0.175468	367982.9831	0.365986	Upper bin = 18.427 ->
7	6	307542	307542	1845252	0.146223	0.146223	306652.4859	2.580235	Lower bin = -8.42578 ->
8	7	218460	218460	1529220	0.104445	0.104445	219037.4899	1.522546	
9	8	136330	136330	1090640	0.065278	0.065278	136898.4312	2.360246	
10	9	76288	76288	686592	0.036266	0.036266	76054.6840	0.715753	
11	10	38211	38211	382110	0.018133	0.018133	38027.3420	0.887000	
12	11	17489	17489	192379	0.008242	0.008242	17285.1555	2.403947	
13	12	7302	7302	87624	0.003434	0.003434	7202.1481	1.384365	
14	13	2775	2775	36075	0.001321	0.001321	2770.0570	0.008821	
15	14	968	968	13552	0.000472	0.000472	989.3061	0.458855	
16	15	340	340	5100	0.000157	0.000157	329.7687	0.317434	
17	16	105	105	1680	0.000049	0.000049	103.0527	0.036796	
TOTAL		2097152	n	5.000598				15.644505	
				media				26.2962	
								p	0.478035

df=k-1

16

Da tabela (5%):

Media

=

5.0006

Variancia

=

5.0074

Desvio_Padrão

=

2.23773

Dataplot

=

0.67 ->

Upper bin

=

18.427 ->

Lower bin

=

-8.42578 ->

Tabela 23 – Goodness of Fit test, Ruído Poisson – lambda = 5.

IMAGE 3D (Lambda = 10)

k	Value (i)	Histogram	O _i	Media	P _{Lambda=10} (i)	F(i)	E _i	Chi-square (X²)
Lambda = 10								
1	0	102	102	0	4.540E-05	4.540E-05	95.2112	0.484060
2	1	941	941	941	4.540E-04	4.540E-04	952.1119	0.129686
3	2	4717	4717	9434	2.270E-03	2.270E-03	4760.5597	0.398576
4	3	15812	15812	47436	7.567E-03	7.567E-03	15868.5323	0.201399
5	4	39812	39812	159248	1.892E-02	1.892E-02	39671.3308	0.498794
6	5	79032	79032	395160	3.783E-02	3.783E-02	79342.6616	1.216378
7	6	132424	132424	794544	6.306E-02	6.306E-02	132237.7694	0.262269
8	7	189276	189276	1324932	9.008E-02	9.008E-02	188911.0991	0.704843
9	8	235640	235640	1885120	1.126E-01	1.126E-01	236138.8739	1.053935
10	9	262272	262272	2360448	1.251E-01	1.251E-01	262376.5265	0.041642
11	10	261832	261832	2618320	1.251E-01	1.251E-01	262376.5265	1.130090
12	11	239128	239128	2630408	1.137E-01	1.137E-01	238524.1150	1.528890
13	12	198072	198072	2376864	9.478E-02	9.478E-02	198770.0958	2.451766
14	13	153263	153263	1992419	7.291E-02	7.291E-02	152900.0737	0.861448
15	14	109536	109536	1533504	5.208E-02	5.208E-02	109214.3384	0.947368
16	15	73198	73198	1097970	3.472E-02	3.472E-02	72809.5589	2.072344
17	16	45288	45288	724608	2.170E-02	2.170E-02	45505.9743	1.044100
18	17	26968	26968	458456	1.276E-02	1.276E-02	26768.2202	1.491021
19	18	14698	14698	264564	7.091E-03	7.091E-03	14871.2334	2.017978
20	19	8021	8021	152399	3.732E-03	3.732E-03	7826.9650	4.810242
21	20	3840	3840	76800	1.866E-03	1.866E-03	3913.4825	1.379762
22	21	1918	1918	40278	8.886E-04	8.886E-04	1863.5631	1.590167
23	22	823	823	18106	4.039E-04	4.039E-04	847.0741	0.684195
24	23	375	375	8625	1.756E-04	1.756E-04	368.2931	0.122138
25	24	143	143	3432	7.317E-05	7.317E-05	153.4555	0.712367
26	26	21	21	546	1.126E-05	1.126E-05	23.6085	0.288220
TOTAL				10.00145	28.123678			
				media				

Media	=	10.00150
Variancia	=	10.10016
Desvio_Padrão	=	3.17807
Dataplot	=	0.95 -->
Upper bin	=	29.06992 -->
Lower bin	=	-9.06692 -->

1
29
0

df=k-1
25
Data tabelas (5%):
37.652

p	0.302191
---	----------

Tabela 24 – Goodness of Fit test, Ruído Poisson – lambda = 10.

IMAGE 3D (Lambda = 20)

k	Value (i)	Histogram	O _i	Media	P _{lambda(i)} Lambda = 20	F(Y)	E _i	Chi-square (X^2) Lambda = 20
	4	72			1.3741E-05			
1	5	184	512	2048	5.4865E-05	0.000252	528.3191	0.504074
	6	256			1.8322E-04			
2	7	1084	1084	7588	5.2347E-04	0.000523	1097.8059	0.173621
3	8	2714	2714	21712	1.3087E-03	0.001309	2744.5147	0.339275
4	9	6084	6084	54756	2.9082E-03	0.002908	6098.9215	0.036507
5	10	12080	12080	120800	5.8164E-03	0.005816	12197.8429	1.138477
6	11	22272	22272	244992	1.0375E-02	0.010575	22177.8963	0.399295
7	12	36640	36640	439680	1.7625E-02	0.017625	36963.1604	2.825318
8	13	56600	56600	735800	2.7116E-02	0.027116	56866.4007	1.248001
9	14	80749	80749	1130486	3.8737E-02	0.038737	81237.7152	2.940046
10	15	108784	108784	1631760	5.1650E-02	0.051650	108316.9537	2.013833
11	16	135856	135856	2173696	6.4562E-02	0.064562	135396.1921	1.561516
12	17	159072	159072	2704224	7.5955E-02	0.075955	159289.6377	0.297359
13	18	177680	177680	3198240	8.4395E-02	0.084395	176988.4864	2.701820
14	19	186553	186553	3544507	8.8837E-02	0.088837	186303.6699	0.333678
15	20	186696	186696	3733920	8.8837E-02	0.088837	186303.6699	0.826194
16	21	177272	177272	3722712	8.4606E-02	0.084606	177432.0665	0.144401
17	22	161382	161382	3550404	7.6915E-02	0.076915	161301.8787	0.039798
18	23	140896	140896	3240608	6.6882E-02	0.066882	140262.5032	2.861194
19	24	116280	116280	2790720	5.5735E-02	0.055735	116885.4193	3.135828

Media	=	19.99630
Variancia	=	19.98117
Desvio_Padrão	=	4.47003
Dataplot	=	1.3 -->
Upper bin	=	46.81648 -->
Lower bin	=	-6.82388 -->

1
47
0

IMAGE 3D (Lambda = 20)

k	Value (i)	Histgrm	O _i	Media	P _{Lambda(i)} Lambda = 20	F(Y)	E _i	Chi-square (X²) Lambda = 20
20	25	93312	93312	2332800	4.4588E-02	0.044588	93508.3355	0.412237
21	26	71808	71808	1867008	3.4299E-02	0.034299	71929.4888	0.205194
22	27	53412	53412	1442124	2.5406E-02	0.025406	53281.1028	0.321579
23	28	38105	38105	1066940	1.8147E-02	0.018147	38057.9306	0.058215
24	29	26032	26032	754928	1.2515E-02	0.012515	26246.8487	1.758686
25	30	17384	17384	521520	8.3436E-03	0.008344	17497.8991	0.741404
26	31	11204	11204	347324	5.3830E-03	0.005383	11288.9672	0.639511
27	32	6976	6976	223232	3.3644E-03	0.003364	7055.6045	0.898133
28	33	4198	4198	138534	2.0390E-03	0.002039	4276.1239	1.427309
29	34	2487	2487	84558	1.1994E-03	0.001199	2515.3670	0.319909
30	35	1420	1420	49700	6.8538E-04	0.000685	1437.3526	0.209491
	36	681			3.8077E-04			
	37	482			2.0582E-04			
31	38	237	1608	57888	1.0833E-04	0.000778	1632.0962	0.355755
	39	143			5.5552E-05			
	40	65			2.7776E-05			
TOTAL		2097152	2097152	19.9953 media				30.867655

df=k-1
30

Da tabela (5%):
43.7729

p
0.421934

Tabela 25 – Goodness of Fit test, Ruído Poisson – lambda = 20.

IMAGE 3D (Lambda = 30)

k	Value (i)	Histgrm	O _i	Media	P _{Lambda=30} (I)	F(Y)	E _i	Chi-square (X²)
Lambda = 30								
1	8	4	12	32	1.5227E-06	6.599E-06	13.8381	0.244148
2	9	8	123	72	5.0758E-06	5.676E-05	119.0268	0.132626
	10	49		490	1.5227E-05			
3	11	74	738	814	4.1529E-05	3.434E-04	720.1905	0.440407
	12	128		1536	1.0382E-04			
4	13	610	3268	7930	2.3959E-04	1.540E-03	3230.0905	0.444921
	14	1468		20552	5.1341E-04			
5	15	1800	11120	27000	1.0268E-03	5.323E-03	11162.8126	0.164199
	16	3968		63488	1.9253E-03			
6	17	7152	30752	121584	3.3976E-03	1.460E-02	30625.8576	0.519557
	18	10496		188928	5.6626E-03			
7	19	20256	68040	384864	8.9409E-03	3.257E-02	68305.4843	1.031863
	20	30720		614400	1.3411E-02			
8	21	37320	126424	783720	1.9159E-02	6.020E-02	126256.3589	0.222591
	22	55168		1213696	2.6126E-02			
9	23	71256	196712	1638888	3.4078E-02	9.371E-02	196531.1248	0.166467
	24	86784		2082816	4.2597E-02			
10	25	109928	261424	2748200	5.1116E-02	1.245E-01	261125.2707	0.341749
	26	124544		3238144	5.8980E-02			
11	27	136880	299016	3695760	6.5534E-02	1.429E-01	299579.8088	1.061088
	28	144384		4042752	7.0215E-02			
12	29	154632	299120	4484328	7.2636E-02	1.429E-01	299743.6032	1.297378
	30	161408		4842240	7.2636E-02			
13	31	137712	263632	4269072	7.0293E-02	1.258E-01	263839.1440	0.162632
	32	132992		4255744	6.5900E-02			
14	33	130640	206080	4311120	5.9909E-02	9.817E-02	205876.8831	0.200394
	34	108160		3677440	5.2861E-02			
15	35	97920	143987	3427200	4.5309E-02	6.837E-02	143386.1867	2.517513
	36	78720		2833920	3.7758E-02			
	37	65267		2414879	3.0614E-02			

Media

Variação

Desvio_Padrão

Dataplot

Upper bin

Lower bin

=

=

=

=

=

=

30.05800

30.24979

5.49998

1.6 -->

63.05788 -->

-2.94188 -->

2

63

0

IMAGE 3D (Lambda = 40)

k	Value (l)	Histgrm	O _i	Media	P _{Lambdas(l)} Lambda = 40	F(V)	E _i	Chi-square (X²/2) Lambda = 40
1	15	11	30	165	3.4885E-06	1.2210E-05	25.6053	0.754257
	16	19		304	8.7211E-06			
2	17	57	142	969	2.0520E-05	6.6121E-05	138.6657	0.080174
	18	85		1530	4.5601E-05			
3	19	245	616	4655	9.6001E-05	2.8800E-04	603.9887	0.238866
	20	371		7420	1.9200E-04			
4	21	768	2128	16128	3.6572E-04	1.0307E-03	2161.4601	0.517975
	22	1360		29920	6.6494E-04			
5	23	2428	6408	55844	1.1564E-03	3.0838E-03	6467.2019	0.541945
	24	3980		95520	1.9274E-03			
6	25	6632	16472	165800	3.0838E-03	7.8281E-03	16416.7434	0.185986
	26	9840		255840	4.7443E-03			
7	27	15744	35912	425088	7.0286E-03	1.7069E-02	35797.2920	0.367568
	28	20168		564704	1.0041E-02			
8	29	29952	67836	868608	1.3849E-02	3.2315E-02	67770.3973	0.063504
	30	37884		1136520	1.8466E-02			
9	31	51072	112144	1583232	2.3827E-02	5.3611E-02	112430.1522	0.728302
	32	61072		1954304	2.9784E-02			
10	33	72832	164159	2403456	3.6102E-02	7.8574E-02	164781.7639	2.353628
	34	91327		3105118	4.2472E-02			
11	35	99584	215412	3485440	4.8540E-02	1.0247E-01	214902.0002	1.210318
	36	115828		4169808	5.3933E-02			
12	37	118528	251023	4385536	5.8306E-02	1.1968E-01	250989.8332	0.004383
	38	132495		5034810	6.1375E-02			
13	39	138880	264864	5416320	6.2949E-02	1.2590E-01	264026.1231	2.658971
	40	125948		5037920	6.2949E-02			
14	41	136064	252212	5578624	6.1413E-02	1.1990E-01	251453.4505	2.288286
	42	116148		4878216	5.8489E-02			
15	43	109312	217024	4700416	5.4408E-02	1.0387E-01	217832.1028	2.997860
	44	107712		4739328	4.9462E-02			
16	45	89472	172192	4026240	4.3966E-02	8.2198E-02	172381.5122	0.208345
	46	82720		3805120	3.8232E-02			

Media	=	39.96720
Variância	=	40.5183
Desvio_Padrão	=	6.36539
Dataplot	=	1.9 -->
Upper bin	=	78.15954 -->
Lower bin	=	1.77486 -->
		2

IMAGE 3D (Lambda = 50)

k	Value (I)	Histgrm	O _i	Media	P _{Lambda(I)} Lambda = 50	F(Y)	E _i	Chi-square (X^2) Lambda = 50
1	21	14	42	294	1.8002E-06	5.8915E-06	12.3554	1.745992
	22	28		616	4.0913E-06			
2	23	51	143	1173	8.8942E-06	2.7424E-05	57.5117	1.912712
	24	92		2208	1.8530E-05			
3	25	157	410	3925	3.7059E-05	1.0833E-04	227.1774	2.935184
	26	253		6578	7.1268E-05			
4	27	410	1060	11070	1.3198E-04	3.6765E-04	771.0176	3.244757
	28	650		18200	2.3567E-04			
5	29	1024	2572	29696	4.0633E-04	1.0836E-03	2272.3774	4.367525
	30	1548		46440	6.7722E-04			
6	31	2372	5790	73532	1.0923E-03	2.7990E-03	5869.9265	2.456190
	32	3418		109376	1.7067E-03			
7	33	4840	11712	159720	2.5859E-03	6.3887E-03	13398.1626	4.522713
	34	6872		233648	3.8028E-03			
8	35	9508	21900	332780	5.4326E-03	1.2978E-02	27216.6002	1.235844
	36	12392		446112	7.5453E-03			
9	37	17352	39860	642024	1.0196E-02	2.3613E-02	49519.1046	2.346765
	38	22508		855304	1.3416E-02			
10	39	28664	64184	1117896	1.7200E-02	3.8701E-02	81161.1198	2.203366
	40	35520		1420800	2.1500E-02			
11	41	43896	96708	1799736	2.6220E-02	5.7434E-02	120448.1711	1.755465
	42	52812		2218104	3.1214E-02			
12	43	62448	135204	2685264	3.6296E-02	7.7541E-02	162614.4419	2.137441
	44	72756		3201264	4.1245E-02			
13	45	83216	174592	3744720	4.5828E-02	9.5641E-02	200572.8546	4.796640
	46	91376		4203296	4.9813E-02			
14	47	103552	212358	4866944	5.2992E-02	1.0819E-01	226896.5600	1.278322
	48	108806		5222688	5.5200E-02			
15	49	114856	230052	5627944	5.6327E-02	1.1265E-01	236252.1450	2.707988
	50	115196		5759800	5.6327E-02			
16	51	114456	230284	5837256	5.5222E-02	1.0832E-01	227165.5240	3.420786
	52	115828		6023056	5.3099E-02			
17	53	112768	211200	5976704	5.0093E-02	9.6475E-02	202323.3316	6.236917
	54	98432		5315328	4.6382E-02			

Media

Variancia

Desvio_Padrão

Dataplot

Upper bin

Lower bin

=

=

=

=

=

=

51.23150

49.58242

7.041478

2.1

93.480368

8.982632

->

->

->

->

->

->

2

92

8

IMAGE 3D (Lambda = 50)

k	Value (i)	Hist,mm	O _i	Media	P _{Lambd(i)} Lambda = 50	F(Y)	E _i	Chi-square (X²) Lambda = 50
18	55	99952	196996	5497360	4.2166E-02	7.9814E-02	167381.6273	2.255049
	56	97044		5434464	3.7648E-02			
19	57	82576	159612	4706832	3.3025E-02	6.1494E-02	128962.3194	3.272931
	58	77036		4468088	2.8469E-02			
20	59	67104	122352	3959136	2.4127E-02	4.4232E-02	92761.6600	1.809167
	60	55248		3314880	2.0106E-02			
21	61	45456	82074	2772816	1.6480E-02	2.9770E-02	62432.7311	2.061227
	62	36618		2270316	1.3290E-02			
22	63	26040	47696	1640520	1.0548E-02	1.8788E-02	39402.0345	2.193178
	64	21656		1385984	8.2405E-03			
23	65	17752	29124	1153880	6.3389E-03	1.1141E-02	23364.3950	2.473369
	66	11372		750552	4.8022E-03			
24	67	9040	15824	605680	3.5837E-03	6.2188E-03	13041.7182	3.694585
	68	6784		461312	2.6351E-03			
25	69	4816	8036	332304	1.9095E-03	3.2734E-03	6864.7848	4.270291
	70	3220		225400	1.3639E-03			
26	71	2048	3473	145408	9.6050E-04	1.6275E-03	3413.1445	1.049671
	72	1425		102600	6.6701E-04			
27	73	1166	1778	85118	4.5686E-04	7.6555E-04	1605.4691	3.276753
	74	612		45288	3.0869E-04			
28	75	452	738	33900	2.0579E-04	3.4118E-04	715.5104	2.523182
	76	286		21736	1.3539E-04			
29	77	174	282	13398	8.7915E-05	1.4427E-04	302.5590	1.249186
	78	108		8424	5.6356E-05			
30	79	64	108	5056	3.5668E-05	5.7961E-05	121.5532	2.225341
	80	44		3520	2.2293E-05			
31	81	23	35	1863	1.3761E-05	2.2152E-05	46.4556	1.225211
	82	12		984	8.3908E-06			
TOTAL		2097152	2097152	51.23153 media	82.823750			
		n			p 7.697E-07			
					Da tabela (5%): 43.773			
					Poisson df=k-1 30			

Tabela 28 – Goodness of Fit test, Ruído Poisson – lambda = 50.

IMAGE 3D (Lambda = 50)

k	Value (l)	Histgm	O _i	Media	P(l)=h(l)/n	P _{Gauss(l)} Lambda = 50	F(V)	E _i	Chi-square (X²2) Lambda = 50
1	21	14	42	294	0.00085	8.22459E-06	17.24821847	48.0059143	Media
	22	28		616	0.00171	1.46664E-05	30.75769583		7.5139
2	23	51	143	1173	0.00311	2.56459E-05	53.7833405	146.0035639	Desvio_Padrão
	24	92		2208	0.00562	4.3974E-05	92.22022342		0.06179
3	25	157	410	3925	0.00958	7.39365E-05	155.0561474	410.7005158	2.1 -->
	26	253		6578	0.01544	0.000121901	255.6443684		0.00119
4	27	410	1060	11070	0.02502	0.000197078	413.3023674	1068.517588	8.982632 -->
	28	650		18200	0.03967	0.000312431	655.2152209		0.06790
5	29	1024	2572	29696	0.06250	0.000485685	1018.554999	2571.190146	
	30	1548		46440	0.09448	0.000740354	1552.635147		0.00026
6	31	2372	5790	73532	0.14478	0.001106646	2320.805599	5722.47839	
	32	3418		109376	0.20862	0.001622044	3401.67279		0.79671
7	33	4840	11712	159720	0.29541	0.002331314	4889.120379	11779.66179	
	34	6872		233648	0.41943	0.003285666	6890.541407		0.38865
8	35	9508	21900	332780	0.58032	0.00454078	9522.706193	22427.52305	
	36	12392		446112	0.75635	0.006153496	12904.81685		0.40799
9	37	17352	39860	642024	1.05908	0.008177072	17148.56277	39493.95753	
	38	22508		855304	1.37378	0.010655115	22345.39476		1.39260
10	39	28664	64184	1117896	1.74951	0.013614536	28551.75165	64325.29059	
	40	35520		1420800	2.16797	0.017058153	35773.53893		0.31034
11	41	43896	96708	1799736	2.67920	0.020957794	43951.67949	96902.59421	
	42	52812		2218104	3.22339	0.025248964	52950.91471		0.39077
12	43	62448	135204	2685264	3.81152	0.029828126	62554.11447	135018.1833	
	44	72756		3201264	4.44067	0.034553561	72464.06886		0.25573
13	45	83216	174592	3744720	5.07910	0.039250398	82314.05062	174001.4497	
	46	91376		4203296	5.57715	0.043719959	91687.39904		0.00429
14	47	103552	212358	4866944	6.32031	0.04775291	100145.1115	207404.2434	
	48	108806		5222688	6.64099	0.05114514	107259.1319		1.31824
15	49	114856	230052	5627944	7.01025	0.05371472	112647.9319	228658.2426	
	50	115196		5759800	7.03101	0.055318027	116010.3107		0.49547
16	51	114456	230284	5837256	6.98584	0.055863028	117153.2602	233163.5709	
	52	115828		6023056	7.06958	0.055318027	116010.3107		0.56271
17	53	112768	211200	5976704	6.88281	0.05371472	112647.9319	219907.0637	
	54	98432		5315328	6.00781	0.05114514	107259.1319		1.74999

IMAGE 3D (Lambda = 50)

k	Value (i)	Histgrm	O _i	Media	P(i)=h(i)/n	P _{exat} (i) Lambda = 50	F(i)	E _i	Chi-square (X ²) Lambda = 50
18	55	99952	196996	5497360	6.10059	0.04775291	100145.1115	191832.5105	1.98386
19	56	97044		5434464	5.92310	0.043719959	91687.39904		
20	57	82576	159612	4706832	5.04004	0.039250398	82314.0562	154778.1195	1.96708
21	58	77036		4468088	4.70190	0.034553561	72464.06886		
22	59	67104	122352	3959136	4.09570	0.029828126	62554.11447	115505.0292	0.87851
23	60	55248		3314880	3.37207	0.025248964	52950.91471		
24	61	45456	82074	2772816	2.77441	0.020957794	43951.67949	79725.21843	0.19736
25	62	36618		2270316	2.23499	0.017058153	35773.53893		
26	63	26040	47696	1640520	1.58936	0.013614536	28551.75165	50897.14642	2.33424
27	64	21656		1385584	1.32178	0.010655115	22345.39476		
28	65	17752	29124	1153880	1.08350	0.008177072	17148.56277	3005.37962	0.74041
29	66	11372		750552	0.69409	0.006153496	12904.81685		
30	67	9040	15824	605680	0.55176	0.00454078	952.706193	16413.2476	0.15442
31	68	6784		461312	0.41406	0.003285666	6890.541407		
32	69	4816	8036	332304	0.29395	0.002331314	4889.120379	8290.793169	1.83032
33	70	3220		225400	0.19653	0.001622044	3401.67279		
34	71	2048	3473	145408	0.12500	0.001106646	2320.805599	3873.440746	1.39802
35	72	1425		102600	0.08698	0.000740354	1552.635147		
36	73	1166	1778	85118	0.07117	0.000485685	1018.554899	1673.770219	1.49064
37	74	612		45288	0.03735	0.000312431	655.2152209		
38	75	452	738	33900	0.02759	0.000197078	413.3023674	668.9467357	1.12815
39	76	286		21736	0.01746	0.000121901	255.6443694		
40	77	174	282	13398	0.01062	7.39365E-05	155.0561474	247.2763708	1.87604
41	78	108		8424	0.00659	4.3974E-05	92.22022342		
42	79	64	108	5056	0.00391	2.56459E-05	53.7833405	84.54103633	0.50954
43	80	44		3520	0.00269	1.46664E-05	30.75769583		
44	81	23	35	1863	0.00140	8.22459E-06	17.24821847	26.73282051	0.55664
45	82	12		984	0.00073	4.52261E-06	9.484602038		
TOTAL		2097152	2097152	51.23153	1.000				26.00128
		n		media					

Gaussiano

df=k-2

29

Da tabela (5%):

42.557

p 0.625423

Tabela 29 – Goodness of Fit test, Ruído Gaussiano – lambda = 50.

TABELA TEORICA DA DISTRIBUIÇÃO QUI-QUADRADO

Chi-Square Distribution Table

The shaded area is equal to α for $\chi^2 = \chi^2_{\alpha}$.

df	$\chi^2_{.995}$	$\chi^2_{.990}$	$\chi^2_{.975}$	$\chi^2_{.950}$	$\chi^2_{.900}$	$\chi^2_{.800}$	$\chi^2_{.700}$	$\chi^2_{.600}$	$\chi^2_{.500}$	$\chi^2_{.400}$
1	0.000	0.000	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.010	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.860
5	0.412	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086	16.750
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	0.989	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090	21.955
9	1.735	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725	26.757
12	3.074	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217	28.300
13	3.565	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688	29.819
14	4.075	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000	34.267
17	5.697	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191	38.582
20	7.434	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566	39.997
21	8.034	8.897	10.283	11.591	13.240	29.615	32.671	35.479	38.932	41.401
22	8.643	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289	42.796
23	9.260	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638	44.181
24	9.886	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.980	45.559
25	10.520	11.524	13.120	14.611	16.473	34.382	37.652	40.646	44.314	46.928
26	11.160	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642	48.290
27	11.808	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963	49.645
28	12.461	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278	50.993
29	13.121	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588	52.336
30	13.787	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892	53.672
40	20.707	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691	66.766
50	27.991	29.707	32.357	34.764	37.689	63.167	67.505	71.420	76.154	79.490
60	35.534	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379	91.952
70	43.275	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425	104.215
80	51.172	53.540	57.153	60.391	64.278	96.578	101.879	106.629	112.329	116.321
90	59.196	61.754	65.647	69.126	73.291	107.565	113.145	118.136	124.116	128.299
100	67.328	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807	140.169

TABELA 30 – Tabela teórica da Distribuição Qui-Quadrado.

ANEXO E

CARACTERÍSTICAS DE OPERAÇÃO DO RECEPTOR (ROC)

Segmentador	Imagens Padrão				Imagens com Ruído Poisson				Imagens Filtradas			
	w1	w2	TP %	FP %	w1	w2	TP %	FP %	w1	w2	TP %	FP %
Genérico	0.9	0.1	99.4945	0.0000	0.9	0.1	91.2380	4.5177	0.9	0.1	96.4712	2.5211
	0.8	0.2	99.5142	0.0000	0.8	0.2	91.2394	4.4322	0.8	0.2	96.7166	2.5782
	0.7	0.3	99.5145	0.0000	0.7	0.3	91.2742	4.3211	0.7	0.3	96.7234	2.6245
	0.6	0.4	99.5347	0.0000	0.6	0.4	92.0069	4.4634	0.6	0.4	96.8319	2.5003
	0.5	0.5	99.5402	0.0000	0.5	0.5	92.1474	4.3249	0.5	0.5	97.7691	2.4356
	0.4	0.6	99.5362	0.0000	0.4	0.6	93.0241	4.2504	0.4	0.6	97.7877	2.5838
	0.3	0.7	99.5102	0.0000	0.3	0.7	92.1023	4.8997	0.3	0.7	97.8192	2.3130
	0.2	0.8	99.5083	0.0000	0.2	0.8	91.7971	4.9354	0.2	0.8	97.8076	2.5545
	0.1	0.9	99.4818	0.0000	0.1	0.9	91.1233	4.9882	0.1	0.9	97.0223	2.7834

TABELA 31 – Resultados da segmentação do VE simulado usando o algoritmo Genérico, considerando diferentes pesos de w_1 e w_2 .

Segmentador	Imagens Padrão				Imagens com Ruído Poisson				Imagens Filtradas			
	w1	w2	TP %	FP %	w1	w2	TP %	FP %	w1	w2	TP %	FP %
LIFO	0.9	0.1	99.5754	0.0000	0.9	0.1	91.2380	4.2177	0.9	0.1	96.4219	2.6778
	0.8	0.2	99.5824	0.0000	0.8	0.2	91.2394	4.2322	0.8	0.2	96.5834	2.5903
	0.7	0.3	99.5991	0.0000	0.7	0.3	91.2742	4.3211	0.7	0.3	97.3780	2.6532
	0.6	0.4	99.6032	0.0000	0.6	0.4	91.5069	4.4634	0.6	0.4	97.6472	2.5076
	0.5	0.5	99.6178	0.0000	0.5	0.5	92.4546	4.4037	0.5	0.5	97.8899	2.3120
	0.4	0.6	99.5972	0.0000	0.4	0.6	93.7263	4.2544	0.4	0.6	97.8932	2.2812
	0.3	0.7	99.5914	0.0000	0.3	0.7	93.1534	4.4474	0.3	0.7	98.2243	2.2131
	0.2	0.8	99.5783	0.0000	0.2	0.8	92.7192	4.7212	0.2	0.8	98.0167	2.3572
	0.1	0.9	99.5772	0.0000	0.1	0.9	92.5271	4.9102	0.1	0.9	97.7567	2.4890

TABELA 32 – Resultados da segmentação do VE simulado usando o algoritmo LIFO, considerando diferentes pesos de w_1 e w_2 .

Segmentador	Imagens Padrão				Imagens com Ruído Poisson				Imagens Filtradas			
	w1	w2	TP %	FP %	w1	w2	TP %	FP %	w1	w2	TP %	FP %
KTetaFOEMS (0.10)	0.9	0.1	99.8775	0.0000	0.9	0.1	91.5385	4.8119	0.9	0.1	97.5489	2.8223
	0.8	0.2	99.8823	0.0000	0.8	0.2	91.7360	4.7383	0.8	0.2	97.6151	2.5873
	0.7	0.3	99.8872	0.0000	0.7	0.3	92.2234	4.3844	0.7	0.3	97.8227	2.5229
	0.6	0.4	99.8936	0.0000	0.6	0.4	92.6945	4.2622	0.6	0.4	98.0273	2.2376
	0.5	0.5	99.8976	0.0000	0.5	0.5	93.2314	4.0172	0.5	0.5	98.1834	2.2171
	0.4	0.6	99.8912	0.0000	0.4	0.6	93.4431	3.9564	0.4	0.6	98.2511	2.1684
	0.3	0.7	99.8885	0.0000	0.3	0.7	93.3285	4.3994	0.3	0.7	98.2957	2.3024
	0.2	0.8	99.8794	0.0000	0.2	0.8	93.0817	4.6345	0.2	0.8	98.3158	2.3566
	0.1	0.9	99.8705	0.0000	0.1	0.9	92.8262	4.8207	0.1	0.9	97.9348	2.6814

TABELA 33 – Resultados da segmentação do VE simulado usando o algoritmo KTetaFOEMS, considerando diferentes pesos de w_1 e w_2 .

ANEXO F

AVALIAÇÃO DA SEGMENTAÇÃO DAS IMAGENS 3D

Neste anexo, a intenção é apresentar com maior detalhe os passos que foram seguidos na obtenção dos resultados no Capítulo 5.

Como é procedimento meramente ilustrativo, vamos tomar como exemplo o Volume N° 1 do grupo de Homens (Caso_H01) como os parâmetros apresentados na Tabela 21 no Anexo A.

Uma vez simulado o volume do ventrículo esquerdo do Caso_H01, são aplicados os algoritmos de segmentação no conjunto volume conformadas por 128 fatias bidimensionais, logo são aplicados os algoritmos de avaliação comparando as imagens Padrões com as imagens após a segmentação. A Tabela 34 mostra a media dos valores das 128 imagens para cada um dos algoritmos utilizados.

RESULTADOS CONSOLIDADOS (1)

Grupo de Imagens : 128 x 128 x 128
Imagens Entrada (IN) : Imagens Padrão
Imagens Saída (OUT) : Imagens Segmentadas

METODO	TP	FP	Max. Distância
Algoritmo Genérico	99.9321	0.00	1.00
Algoritmo LIFO	100.0000	0.0000	0.00
Algoritmo kTetaFOEMS	Teta = 0.10	99.9976	0.0000
	Teta = 0.15	99.9976	0.0000
	Teta = 0.20	99.9976	0.0000
	Teta = 0.25	99.9976	0.0000
	Teta = 0.30	99.9976	0.0000
	Teta = 0.35	99.9976	0.0000
Algoritmo de Pesos Dinâmicos	99.9821	0.0000	1.00

TABELA 34 – Resultados parciais do processo de segmentação das Imagens Padrões do primeiro volume do grupo de Homens (Caso_H01).

O segundo passo consiste em pegar as imagens padrões e adicionar o ruído Poisson em cada uma das fatias que conformam o volume testado. Neste passo, pegamos as imagens com ruidosas e aplicamos os algoritmos de segmentação. Logo obtemos os parâmetros de avaliação através dos algoritmos avaliadores os quais comparam as imagens Padrões com as imagens segmentadas. A Tabela 35 mostra este conjunto de valores que foram obtidas fazendo a media dos valores das 128 imagens para cada um dos algoritmos utilizados.

É válido aclarar que no caso do algoritmo kTetaFOEMS, no qual foram utilizadas seis diferentes valores de Teta, foi escolhido o conjunto de valores que apresentou o número maior de Verdadeiros Positivos (VP).

RESULTADOS CONSOLIDADOS (2)

Grupo de Imagens : 128 x 128 x 128
 Imagens Entrada (IN) : Imagens com Ruído Poisson
 Imagens Saida (OUT) : Imagens Segmentadas

METODO	TP	FP	Max. Distância
Algoritmo Genérico	92.7154	4.5569	4
Algoritmo LIFO	93.7623	4.2034	4
Algoritmo kTetaFOEMS	Teta = 0.10	92.8012	4.1032
	Teta = 0.15	92.8845	4.1247
	Teta = 0.20	92.0460	4.4423
	Teta = 0.25	91.7561	4.5278
	Teta = 0.30	91.5078	4.5483
	Teta = 0.35	91.5134	4.7245
Algoritmo de Pesos Dinâmicos	92.7944	4.5123	5

TABELA 35 – Resultados parciais do processo de segmentação das Imagens com Ruído Poisson do primeiro volume do grupo de Homens (Caso_H01).

Finalmente, é aplicada a filtragem no volume com ruído e sobre o volume resultante são aplicados os algoritmos de segmentação. Na sequência, são obtidos os parâmetros de avaliação através dos algoritmos de avaliação, comparando as imagens Padrões com as imagens obtidas após o processo de segmentação. A Tabela 36 mostra este conjunto de valores que foram obtidas fazendo a media dos valores das 128 imagens para cada um dos algoritmos utilizados.

Como foi indicado no passo anterior, para o caso do algoritmo kTetaFOEMS, foram escolhidos os valores que apresentaram o número maior de Verdadeiros Positivos (VP).

RESULTADOS CONSOLIDADOS (3)

Grupo de Imagens : 128 x 128 x 128
Imagens Entrada (IN) : Imagens Filtradas
Imagens Saida (OUT) : Imagens Segmentadas

METODO	TP	FP	Max. Distância	
Algoritmo Genérico	97.103	2.7284	3	
Algoritmo LIFO	97.7624	2.3245	3	
Algoritmo kTetaFOEMS	Teta = 0.10	97.8865	2.7823	2
	Teta = 0.15	97.3455	2.5467	2
	Teta = 0.20	97.4594	2.8034	3
	Teta = 0.25	97.205	2.8242	3
	Teta = 0.30	97.237	2.7424	3
	Teta = 0.35	96.3721	2.8893	3
Algoritmo de Pesos Dinâmicos	97.5368	2.3156	2	

TABELA 36 – Resultados parciais do processo de segmentação das Imagens Filtradas do primeiro volume do grupo de Homens (Caso_H01).

ANEXO G

COMPARATIVO DOS TEMPOS DE SEGMENTAÇÃO

Na Tabela 37 apresenta um comparativo dos tempos computacionais utilizados nas diferentes etapas dentro da sequência de processamento implementada. Estes valores representam os tempos obtidos na análise de um dos volumes do VE simulado.

Algoritmo	Média dos tempos (em ms)		
	Imagens Padrões	Imagens com Ruído	Imagens Filtradas
Genérico	1640	156594	15125
LIFO	1499	6101	1252
KFOEMS ($\theta=0.10$)	1048	3849	1082
Pesos Dinâmicos	2004	22340	27138

TABELA 37 – Comparativo das médias dos tempos utilizados no processo de segmentação do VE simulado para os quatro algoritmos implementados.

ANEXO H

LISTAGEM DE PROGRAMAS

H.1 – Simulação do Ventrículo Esquerdo do Coração (3D)

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ventrículo Esquerdo Simulado (VES_versão 3.2)
% AUTOR: Edward Flórez Pacheco
% DATA: 26/mar/2011
% Escola Politécnica da Universidade de São Paulo
% Laboratorio de Processamento de Imagens Médicas
%
% OBJETIVO: Simular o Ventrículo Esquerdo do Coração
% gerando cortes (fatias) ao longo do eixo "Z"
% SEM adição de ruído quântico tipo 'Poisson'
% (128x128x128)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

largura=128;
altura=128;
profundidade=128;
centroX=largura/2;
centroY=altura/2;
centroZ=profundidade/20;
raioX=30;
raioY=30;
raioZ=40;

cubo=ones(largura,altura,profundidade, 'uint8')*2;

%% Criação do objeto 3D

for i=1:largura
    for j=1:altura
        for k=1:profundidade

            if ((i-centroX)/raioX)^2+((j-
centroY)/raioY)^2<((k-centroZ+10)/raioZ) && ((i-
centroX)/raioX)^2+((j-centroY)/raioY)^2>((k-centroZ-10)/raioZ)
                cubo(i,j,k)=20;
            end
        end
    end
end

%% Gravação das imagens

distanciaCortes=1;
contaImagem=1;

```

```

for z=1:distanciaCortes:profundidade
    imshow(cubo(:,:,z));

    if (contaImagem < 10)

imwrite(cubo(:,:,z),['F:\2011\IMAGENS.2011_CinC\Grupo_1\Caso_H02\ (
a) Imagens_Padrao\Image00',num2str(contaImagem),'.png']);

        elseif (contaImagem < 100)

imwrite(cubo(:,:,z),['F:\2011\IMAGENS.2011_CinC\Grupo_1\Caso_H02\ (
a) Imagens_Padrao\Image0',num2str(contaImagem),'.png']);

        elseif (contaImagem < 129)

imwrite(cubo(:,:,z),['F:\2011\IMAGENS.2011_CinC\Grupo_1\Caso_H02\ (
a) Imagens_Padrao\Image',num2str(contaImagem),'.png']);

        end

        pause(.1);
        contaImagem=contaImagem+1;
end

```

H.2 – Simulação do VE (3D) com incorporação do Ruído Poisson

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ventrículo Esquerdo Simulado
% @AUTOR: Edward Flórez Pacheco
% @VERSÃO: 4.2
% @DATA: 30/Março/2011
% Escola Politecnica da Universidade de São Paulo (EPUSP) - Brazil
% Laboratorio de Engenharia Biomedica (LEB)
%
% OBJETIVO: Simular o Ventrículo Esquerdo do Coração
% gerando cortes (fatias) ao longo do eixo "Z",
% além disso, adiciona ruído quântico tipo 'Poisson'
% (128x128x128)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

largura=128;
altura=128;
profundidade=128;
centroX=largura/2;
centroY=altura/2;
centroZ=profundidade/20;
raioX=30;
raioY=30;
raioZ=40;

```

```

cubo=ones(largura,altura,profundidade, 'uint8')*2;

%% Criação do objeto 3D

for i=1:largura
    for j=1:altura
        for k=1:profundidade

            if ((i-centroX)/raioX)^2+((j-
centroY)/raioY)^2<((k-centroZ+10)/raioZ) && ((i-
centroX)/raioX)^2+((j-centroY)/raioY)^2>((k-centroZ-10)/raioZ)
                cubo(i,j,k)=20;
            end
        end
    end
end

%% Gravação das imagens

distanciaCortes=1;
contaImagem=1;
for z=1:distanciaCortes:profundidade
    I=cubo(:, :, z);
    J=imnoise(I, 'poisson');
    imshow(J);

    if (contaImagem < 10)

        imwrite(J, ['I:\USP_Eng.Biomedica\MESTRADO 2010-
2011\TESES_MESTRADO\Teste_Imagens2011\Imagens_128X128X128\ (b) Image
ns_RuidoPoisson\Image00', num2str(contaImagem), '.png']);

        elseif (contaImagem < 100)

            imwrite(J, ['I:\USP_Eng.Biomedica\MESTRADO 2010-
2011\TESES_MESTRADO\Teste_Imagens2011\Imagens_128X128X128\ (b) Image
ns_RuidoPoisson\Image0', num2str(contaImagem), '.png']);

            elseif (contaImagem < 129)

                imwrite(J, ['I:\USP_Eng.Biomedica\MESTRADO 2010-
2011\TESES_MESTRADO\Teste_Imagens2011\Imagens_128X128X128\ (b) Image
ns_RuidoPoisson\Image', num2str(contaImagem), '.png']);
            end

            pause(.1);
            contaImagem=contaImagem+1;
        end
    end
end

```


H.3 – Filtro da Mediana (3D)

```

import ij.*;
import ij.plugin.filter.PlugInFilter;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import java.util.*;
import java.text.NumberFormat;
import java.lang.Math.*;

/*****
* Bloco adicional de Filtragem (3D)
* Algoritmo utilizando o Filtro da Mediana
* Ref.: Baseado no algoritmo do C. Philip & V. Bindokas (2004)
* Caracteristicas do algoritmo:
* 1) Permite seleccionar o tamanho da janela: 3x3, 5x5 e 7x7.
* 2) Permite seleccionar multiples repetiçõs de filtrado.
* 3) Trabalha no filtrado de imagens 3D (stack).
*
* @author Edward Flórez Pacheco
* @versão 1.2
* @data 27/out/2011
* Escola Politécnica da Universidade de São Paulo - Brasil
* Laboratório de Engenharia Biomedica (LEB)
*****/

public class Median_Filter_3D implements PlugInFilter{
    private ImagePlus imp;
    private ImagePlus imp2;
    private ImagePlus impcopy;
    private ImageStack stack;
    private ImageStack stack2;
    private ImageStack tempstack;
    private ImageProcessor ip;
    private ImageProcessor ip2;
    private boolean atebit = false;
    private boolean include = false;
    private boolean astack = true;
    private NumberFormat nf = NumberFormat.getInstance();
    private double prognum;
    private String progstr;
    private double nsize = 1;
    private double times = 1;
    private String otitle;
    private String titlestring;
    private ImageWindow win;
    private String sizes[] = {"3x3", "5x5", "7x7"};
    private boolean quit = false;

    public int setup(String argv, ImagePlus imp){
        try{stack = imp.getStack();}
        catch(Exception e){
            IJ.showMessage("Uma grupo de imagens que conformam um
stack (3D) deve ser seleccionado!!");
            return DONE;

```

```

    }
    otitle = imp.getTitle();
    GenericDialog d = new GenericDialog("Filtro da Mediana
(3D)", IJ.getInstance());
    d.addChoice("Tamanho da janela (Kernel):", sizes, sizes[0]);
    d.addNumericField("Número de repetições:", times, 0);
    //d.addCheckbox("Toma em conta o pixel central", include);
    d.showDialog();
    if(d.wasCanceled()) return DONE;
    nsize = Math.floor(d.getNextChoiceIndex());
    times = d.getNextNumber();
    //include = d.getNextBoolean();
    String ErrorMessage =
        new String("O número ingressado é invalido.\nO valor tem
que ser maior ou igual a 1.\nUma repetição sera executada por
defeito!!");
    if(d.invalidNumber() || times < 1) {
        IJ.showMessage("Error", ErrorMessage);
        times = 1;
    }
    nf.setMaximumFractionDigits(2);
    nf.setMinimumFractionDigits(2);
    this.imp = imp;
    if(imp.getBitDepth() == 8) atebit = true;
    win = imp.getWindow();
    if (win != null) win.running = true;
    if(imp.getStack().getSize() == 1) astack = false;
    return DOES_8G + DOES_16 + STACK_REQUIRED + NO_UNDO;
}

public void run(ImageProcessor ip){
    imp2 = Hybrid3dMedianizer(imp);
    if(quit) return;
    imp2.getProcessor().resetMinAndMax();
    imp2.show();
}

private ImagePlus Hybrid3dMedianizer(ImagePlus imp){
    int m = stack.getWidth();
    int n = stack.getHeight();
    imp2 = new ImagePlus("Filtro Mediana (3D)", imp.getStack());
    stack2 = imp2.createEmptyStack();
    int dimension = m*n;
    int stacksize = stack.getSize();

    short beforeslice[];
    short thisslice[];
    short afterslice[];
    short newslice[];

    byte beforeslice8[];
    byte thisslice8[];
    byte afterslice8[];
    byte newslice8[];

    if(atebit) { // 8bit scenario

```

```

        beforeslice8 = new byte[dimension];
        thisslice8 = new byte[dimension];
        afterslice8 = new byte[dimension];
        newslice8 = new byte[dimension];
    }
    else{//16bit scenario
        beforeslice = new short[dimension];
        thisslice = new short[dimension];
        afterslice = new short[dimension];
        newslice = new short[dimension];
    }
    double dubbeforeslice[] = new double[dimension];
    double dubthisslice[] = new double[dimension];
    double dubafterslice[] = new double[dimension];
    double filteredslice[] = new double[dimension];

    double marraythisP[] = new double[5];
    double marraythisX[] = new double[5];
    double marray3P[] = new double[7];
    double marray3Xa[] = new double[5];
    double marray3Xb[] = new double[5];
    double marray3Xc[] = new double[5];
    double marray3Xd[] = new double[5];
    double medianarray[] = new double[7];
    if(include)medianarray = new double[8];

    if(nsize==1){
        marraythisP = new double[9];
        marraythisX = new double[9];
    }
    if(nsize==2){
        marraythisP = new double[13];
        marraythisX = new double[13];
    }

    for(int i=1;i<=stacksize;++i){

        if(atebit){
            try{beforeslice8 = (byte[])stack.getPixels(i-
1);}catch(Exception e){beforeslice8=(byte[])stack.getPixels(i);}
            thisslice8 = (byte[])stack.getPixels(i);
            try{afterslice8 =
(byte[])stack.getPixels(i+1);}catch(Exception
e){afterslice8=(byte[])stack.getPixels(i);}
            dubbeforeslice = byte2double(beforeslice8);
            dubthisslice = byte2double(thisslice8);
            dubafterslice = byte2double(afterslice8);
        }
        else{
            try{beforeslice = (short[])stack.getPixels(i-
1);}catch(Exception e){beforeslice=(short[])stack.getPixels(i);}
            thisslice = (short[])stack.getPixels(i);
            try{afterslice =
(short[])stack.getPixels(i+1);}catch(Exception
e){afterslice=(short[])stack.getPixels(i);}

```

```

        dubbeforeslice = short2double(beforeslice);
        dubthisslice = short2double(thisslice);
        dubafterslice = short2double(afterslice);
    }

    if(nsize==0) {
        for(int k=0;k<times;++k) {
            for(int j=0;j<dimension;++j) {
                try{marraythisP[0] = dubthisslice[(j-
m)];}catch(Exception e){marraythisP[0]=dubthisslice[j];}
                try{marraythisP[1] = dubthisslice[(j-
1)];}catch(Exception e){marraythisP[1]=dubthisslice[j];}
                try{marraythisP[2] = dubthisslice[j];}
            }catch(Exception e){marraythisP[2]=dubthisslice[j];}
                try{marraythisP[3] =
dubthisslice[(j+1)];}catch(Exception
e){marraythisP[3]=dubthisslice[j];}
                try{marraythisP[4] =
dubthisslice[(j+m)];}catch(Exception
e){marraythisP[4]=dubthisslice[j];}

                try{marraythisX[0] = dubthisslice[(j-
(m+1))];}catch(Exception e){marraythisX[0]=dubthisslice[j];}
                try{marraythisX[1] = dubthisslice[(j-(m-
1))];}catch(Exception e){marraythisX[1]=dubthisslice[j];}
                try{marraythisX[2] = dubthisslice[j];}
            }catch(Exception e){marraythisX[2]=dubthisslice[j];}
                try{marraythisX[3] = dubthisslice[(j+(m-
1))];}catch(Exception e){marraythisX[3]=dubthisslice[j];}
                try{marraythisX[4] =
dubthisslice[(j+(m+1))];}catch(Exception
e){marraythisX[4]=dubthisslice[j];}

                try{marray3P[0] = dubbeforeslice[j];}
            }catch(Exception e){marray3P[0]=dubthisslice[j];}
                try{marray3P[1] = dubthisslice[(j-
m)];}catch(Exception e){marray3P[1]=dubthisslice[j];}
                try{marray3P[2] = dubthisslice[(j-
1)];}catch(Exception e){marray3P[2]=dubthisslice[j];}
                try{marray3P[3] = dubthisslice[j];}
            }catch(Exception e){marray3P[3]=dubthisslice[j];}
                try{marray3P[4] =
dubthisslice[(j+1)];}catch(Exception
e){marray3P[4]=dubthisslice[j];}
                try{marray3P[5] =
dubthisslice[(j+m)];}catch(Exception
e){marray3P[5]=dubthisslice[j];}
                try{marray3P[6] = dubafterslice[j];}
            }catch(Exception e){marray3P[6]=dubthisslice[j];}

                try{marray3Xa[0] = dubbeforeslice[(j-
(m+1))];}catch(Exception e){marray3Xa[0]=dubthisslice[j];}
                try{marray3Xa[1] = dubafterslice[(j+(m+1))];}
            }catch(Exception e){marray3Xa[1]=dubthisslice[j];}
                try{marray3Xa[2] = dubthisslice[j];}
            }catch(Exception e){marray3Xa[2]=dubthisslice[j];}

```

```

        try{marray3Xa[3] = dubbeforeslice[(j+(m-
1))];}catch(Exception e){marray3Xa[3]=dubthisslice[j];}
        try{marray3Xa[4] = dubafterslice[(j-(m-1))];}
    }catch(Exception e){marray3Xa[4]=dubthisslice[j];}

        try{marray3Xb[0] = dubbeforeslice[(j-
m)];}catch(Exception e){marray3Xb[0]=dubthisslice[j];}
        try{marray3Xb[1] = dubafterslice[(j+m)];}
    }catch(Exception e){marray3Xb[1]=dubthisslice[j];}
        try{marray3Xb[2] = dubthisslice[j];}
    }catch(Exception e){marray3Xb[2]=dubthisslice[j];}
        try{marray3Xb[3] =
dubbeforeslice[(j+m)];}catch(Exception
e){marray3Xb[3]=dubthisslice[j];}
        try{marray3Xb[4] = dubafterslice[(j-m)];}
    }catch(Exception e){marray3Xb[4]=dubthisslice[j];}

        try{marray3Xc[0] = dubbeforeslice[(j-(m-
1))];}catch(Exception e){marray3Xc[0]=dubthisslice[j];}
        try{marray3Xc[1] = dubafterslice[(j+(m-1))];}
    }catch(Exception e){marray3Xc[1]=dubthisslice[j];}
        try{marray3Xc[2] = dubthisslice[j];}
    }catch(Exception e){marray3Xc[2]=dubthisslice[j];}
        try{marray3Xc[3] = dubbeforeslice[(j+(m-
1))];}catch(Exception e){marray3Xc[3]=dubthisslice[j];}
        try{marray3Xc[4] = dubafterslice[(j-(m-1))];}
    }catch(Exception e){marray3Xc[4]=dubthisslice[j];}

        try{marray3Xd[0] = dubbeforeslice[(j-
1)];}catch(Exception e){marray3Xd[0]=dubthisslice[j];}
        try{marray3Xd[1] = dubafterslice[(j+1)];}
    }catch(Exception e){marray3Xd[1]=dubthisslice[j];}
        try{marray3Xd[2] = dubthisslice[j];}
    }catch(Exception e){marray3Xd[2]=dubthisslice[j];}
        try{marray3Xd[3] =
dubbeforeslice[(j+1)];}catch(Exception
e){marray3Xd[3]=dubthisslice[j];}
        try{marray3Xd[4] = dubafterslice[(j-1)];}
    }catch(Exception e){marray3Xd[4]=dubthisslice[j];}

        medianarray[0] = median(marraythisX);
        medianarray[1] = median(marraythisP);
        medianarray[2] = median(marray3P);
        medianarray[3] = median(marray3Xa);
        medianarray[4] = median(marray3Xb);
        medianarray[5] = median(marray3Xc);
        medianarray[6] = median(marray3Xd);

        if(include)medianarray[7]= dubthisslice[j];
        filteredslice[j] = median(medianarray);

    }

    for(int
h=0;h<dimension;++h)dubthisslice[h]=filteredslice[h]; //Revisar
}

```

```

    }

    if(nsize==1){
        for(int k=0;k<times;++k){
            for(int j=0;j<dimension;++j){

                try{marraythisP[0] = dubthisslice[(j-
m)]]; }catch(Exception e){marraythisP[0]=dubthisslice[j];}
                try{marraythisP[1] = dubthisslice[(j-
1)]]; }catch(Exception e){marraythisP[1]=dubthisslice[j];}
                try{marraythisP[2] = dubthisslice[j];
}catch(Exception e){marraythisP[2]=dubthisslice[j];}
                try{marraythisP[3] =
dubthisslice[(j+1)]]; }catch(Exception
e){marraythisP[3]=dubthisslice[j];}
                try{marraythisP[4] =
dubthisslice[(j+m)]]; }catch(Exception
e){marraythisP[4]=dubthisslice[j];}

                try{marraythisP[5] = dubthisslice[(j-
2*m)]]; }catch(Exception e){
                    try{marraythisP[5]=dubthisslice[j-
m]; }catch(Exception ee){marraythisP[5]=dubthisslice[j];}}

                try{marraythisP[6] = dubthisslice[(j-
2)]]; }catch(Exception e){
                    try{marraythisP[6]=dubthisslice[j-
1]; }catch(Exception ee){marraythisP[6]=dubthisslice[j];}}

                try{marraythisP[7] =
dubthisslice[(j+2)]]; }catch(Exception e){

try{marraythisP[7]=dubthisslice[j+1]; }catch(Exception
ee){marraythisP[7]=dubthisslice[j];}}

                try{marraythisP[8] =
dubthisslice[(j+2*m)]]; }catch(Exception e){

try{marraythisP[8]=dubthisslice[j+m]; }catch(Exception
ee){marraythisP[8]=dubthisslice[j];}}

                try{marraythisX[0] = dubthisslice[(j-
(m+1) )]]; }catch(Exception e){marraythisX[0]=dubthisslice[j];}
                try{marraythisX[1] = dubthisslice[(j-(m-
1) )]]; }catch(Exception e){marraythisX[1]=dubthisslice[j];}
                try{marraythisX[2] = dubthisslice[j];
}catch(Exception e){marraythisX[2]=dubthisslice[j];}
                try{marraythisX[3] = dubthisslice[(j+(m-
1) )]]; }catch(Exception e){marraythisX[3]=dubthisslice[j];}
                try{marraythisX[4] =
dubthisslice[(j+(m+1) )]]; }catch(Exception
e){marraythisX[4]=dubthisslice[j];}

                try{marraythisX[5] = dubthisslice[(j-
(2*m+2) )]]; }catch(Exception e){

```

```

        try{marraythisX[5]=dubthisslice[j-(
(m+1))];}catch(Exception ee){marraythisX[5]=dubthisslice[j];}}

        try{marraythisX[6] = dubthisslice[(j-(2*m-
2))];}catch(Exception e){
            try{marraythisX[6]=dubthisslice[j-(m-
1)];}catch(Exception ee){marraythisX[6]=dubthisslice[j];}}

        try{marraythisX[7] = dubthisslice[(j+(2*m-
2))];}catch(Exception e){
            try{marraythisX[7]=dubthisslice[j+(m-
1)];}catch(Exception ee){marraythisX[7]=dubthisslice[j];}}

        try{marraythisX[8] =
dubthisslice[(j+(2*m+2))];}catch(Exception e){

try{marraythisX[8]=dubthisslice[j+(m+1)];}catch(Exception
ee){marraythisX[8]=dubthisslice[j];}}

        try{marray3P[0] = dubbeforeslice[j];
}catch(Exception e){marray3P[0]=dubthisslice[j];}
        try{marray3P[1] = dubthisslice[(j-
m)];}catch(Exception e){marray3P[1]=dubthisslice[j];}
        try{marray3P[2] = dubthisslice[(j-
1)];}catch(Exception e){marray3P[2]=dubthisslice[j];}
        try{marray3P[3] = dubthisslice[j];
}catch(Exception e){marray3P[3]=dubthisslice[j];}
        try{marray3P[4] =
dubthisslice[(j+1)];}catch(Exception
e){marray3P[4]=dubthisslice[j];}
        try{marray3P[5] =
dubthisslice[(j+m)];}catch(Exception
e){marray3P[5]=dubthisslice[j];}
        try{marray3P[6] = dubafterslice[j];
}catch(Exception e){marray3P[6]=dubthisslice[j];}

        try{marray3Xa[0] = dubbeforeslice[(j-
(m+1))];}catch(Exception e){marray3Xa[0]=dubthisslice[j];}
        try{marray3Xa[1] = dubafterslice[(j+(m+1))];
}catch(Exception e){marray3Xa[1]=dubthisslice[j];}
        try{marray3Xa[2] = dubthisslice[j];
}catch(Exception e){marray3Xa[2]=dubthisslice[j];}
        try{marray3Xa[3] = dubbeforeslice[(j+(m-
1))];}catch(Exception e){marray3Xa[3]=dubthisslice[j];}
        try{marray3Xa[4] = dubafterslice[(j-(m-1))];
}catch(Exception e){marray3Xa[4]=dubthisslice[j];}

        try{marray3Xb[0] = dubbeforeslice[(j-
m)];}catch(Exception e){marray3Xb[0]=dubthisslice[j];}
        try{marray3Xb[1] = dubafterslice[(j+m)];
}catch(Exception e){marray3Xb[1]=dubthisslice[j];}
        try{marray3Xb[2] = dubthisslice[j];
}catch(Exception e){marray3Xb[2]=dubthisslice[j];}
        try{marray3Xb[3] =
dubbeforeslice[(j+m)];}catch(Exception
e){marray3Xb[3]=dubthisslice[j];}

```

```

        try{marray3Xb[4] = dubafterslice[(j-m)];
    }catch(Exception e){marray3Xb[4]=dubthisslice[j];}

        try{marray3Xc[0] = dubbeforeslice[(j-(m-
1))];}catch(Exception e){marray3Xc[0]=dubthisslice[j];}
        try{marray3Xc[1] = dubafterslice[(j+(m-1))];
    }catch(Exception e){marray3Xc[1]=dubthisslice[j];}
        try{marray3Xc[2] = dubthisslice[j];
    }catch(Exception e){marray3Xc[2]=dubthisslice[j];}
        try{marray3Xc[3] = dubbeforeslice[(j+(m-
1))];}catch(Exception e){marray3Xc[3]=dubthisslice[j];}
        try{marray3Xc[4] = dubafterslice[(j-(m-1))];
    }catch(Exception e){marray3Xc[4]=dubthisslice[j];}

        try{marray3Xd[0] = dubbeforeslice[(j-
1)];}catch(Exception e){marray3Xd[0]=dubthisslice[j];}
        try{marray3Xd[1] = dubafterslice[(j+1)];
    }catch(Exception e){marray3Xd[1]=dubthisslice[j];}
        try{marray3Xd[2] = dubthisslice[j];
    }catch(Exception e){marray3Xd[2]=dubthisslice[j];}
        try{marray3Xd[3] =
dubbeforeslice[(j+1)];}catch(Exception
e){marray3Xd[3]=dubthisslice[j];}
        try{marray3Xd[4] = dubafterslice[(j-1)];
    }catch(Exception e){marray3Xd[4]=dubthisslice[j];}

        medianarray[0] = median(marraythisX);
        medianarray[1] = median(marraythisP);
        medianarray[2] = median(marray3P);
        medianarray[3] = median(marray3Xa);
        medianarray[4] = median(marray3Xb);
        medianarray[5] = median(marray3Xc);
        medianarray[6] = median(marray3Xd);

        if(include)medianarray[7]= dubthisslice[j];
        filteredslice[j] = median(medianarray);

    }
    for(int
h=0;h<dimension;++h)dubthisslice[h]=filteredslice[h];
    }
}

if(nsize==2){
    for(int k=0;k<times;++k){
        for(int j=0;j<dimension;++j){

            try{marraythisP[0] = dubthisslice[(j-
m)];}catch(Exception e){marraythisP[0]=dubthisslice[j];}
            try{marraythisP[1] = dubthisslice[(j-
1)];}catch(Exception e){marraythisP[1]=dubthisslice[j];}
            try{marraythisP[2] = dubthisslice[j];
        }catch(Exception e){marraythisP[2]=dubthisslice[j];}
            try{marraythisP[3] =
dubthisslice[(j+1)];}catch(Exception
e){marraythisP[3]=dubthisslice[j];}

```



```

        try{marraythisP[4] =
dubthisslice[(j+m)];}catch(Exception
e){marraythisP[4]=dubthisslice[j];}

        try{marraythisP[5] = dubthisslice[(j-
2*m)];}catch(Exception e){
            try{marraythisP[5]=dubthisslice[j-
m];}catch(Exception ee){marraythisP[5]=dubthisslice[j];}}

        try{marraythisP[6] = dubthisslice[(j-
2)];}catch(Exception e){
            try{marraythisP[6]=dubthisslice[j-
1];}catch(Exception ee){marraythisP[6]=dubthisslice[j];}}

        try{marraythisP[7] =
dubthisslice[(j+2)];}catch(Exception e){

try{marraythisP[7]=dubthisslice[j+1];}catch(Exception
ee){marraythisP[7]=dubthisslice[j];}}

        try{marraythisP[8] =
dubthisslice[(j+2*m)];}catch(Exception e){

try{marraythisP[8]=dubthisslice[j+m];}catch(Exception
ee){marraythisP[8]=dubthisslice[j];}}

        try{marraythisP[9] = dubthisslice[(j-
3*m)];}catch(Exception e){
            try{marraythisP[9] = dubthisslice[(j-
2*m)];}catch(Exception ee){
                try{marraythisP[9]=dubthisslice[j-
m];}catch(Exception eee){marraythisP[9]=dubthisslice[j];}}}

        try{marraythisP[10] = dubthisslice[(j-
3)];}catch(Exception e){
            try{marraythisP[10] = dubthisslice[(j-
2)];}catch(Exception ee){
                try{marraythisP[10]=dubthisslice[j-
1];}catch(Exception eee){marraythisP[10]=dubthisslice[j];}}}

        try{marraythisP[11] =
dubthisslice[(j+3)];}catch(Exception e){
            try{marraythisP[11] =
dubthisslice[(j+2)];}catch(Exception ee){

                try{marraythisP[11]=dubthisslice[j+1];}catch(Exception
eee){marraythisP[11]=dubthisslice[j];}}}

        try{marraythisP[12] =
dubthisslice[(j+3*m)];}catch(Exception e){
            try{marraythisP[12] =
dubthisslice[(j+2*m)];}catch(Exception ee){

                try{marraythisP[12]=dubthisslice[j+m];}catch(Exception
eee){marraythisP[12]=dubthisslice[j];}}}

```

```

        try{marraythisX[0] = dubthisslice[(j-
(m+1))];}catch(Exception e){marraythisX[0]=dubthisslice[j];}
        try{marraythisX[1] = dubthisslice[(j-(m-
1))];}catch(Exception e){marraythisX[1]=dubthisslice[j];}
        try{marraythisX[2] = dubthisslice[j];}
    }catch(Exception e){marraythisX[2]=dubthisslice[j];}
        try{marraythisX[3] = dubthisslice[(j+(m-
1))];}catch(Exception e){marraythisX[3]=dubthisslice[j];}
        try{marraythisX[4] =
dubthisslice[(j+(m+1))];}catch(Exception
e){marraythisX[4]=dubthisslice[j];}

        try{marraythisX[5] = dubthisslice[(j-
(2*m+2))];}catch(Exception e){
        try{marraythisX[5]=dubthisslice[j-
(m+1)];}catch(Exception ee){marraythisX[5]=dubthisslice[j];}}

        try{marraythisX[6] = dubthisslice[(j-(2*m-
2))];}catch(Exception e){
        try{marraythisP[6]=dubthisslice[j-(m-
1)];}catch(Exception ee){marraythisX[6]=dubthisslice[j];}}

        try{marraythisX[7] = dubthisslice[(j+(2*m-
2))];}catch(Exception e){
        try{marraythisX[7]=dubthisslice[j+(m-
1)];}catch(Exception ee){marraythisX[7]=dubthisslice[j];}}

        try{marraythisX[8] =
dubthisslice[(j+(2*m+2))];}catch(Exception e){

try{marraythisX[8]=dubthisslice[j+(m+1)];}catch(Exception
ee){marraythisX[8]=dubthisslice[j];}}

        try{marraythisX[9] = dubthisslice[(j-
(3*m+3))];}catch(Exception e){
        try{marraythisX[9] = dubthisslice[(j-
(2*m+2))];}catch(Exception ee){
        try{marraythisX[9]=dubthisslice[j-
(m+1)];}catch(Exception eee){marraythisX[9]=dubthisslice[j];}}}

        try{marraythisX[10] = dubthisslice[(j-(3*m-
3))];}catch(Exception e){
        try{marraythisX[10] = dubthisslice[(j-(2*m-
2))];}catch(Exception ee){
        try{marraythisX[10]=dubthisslice[j-(m-
1)];}catch(Exception eee){marraythisX[10]=dubthisslice[j];}}}

        try{marraythisX[11] = dubthisslice[(j+(3*m-
3))];}catch(Exception e){
        try{marraythisX[11] = dubthisslice[(j+(3*m-
3))];}catch(Exception ee){
        try{marraythisX[11]=dubthisslice[j+(m-
1)];}catch(Exception eee){marraythisX[11]=dubthisslice[j];}}}

```

```

        try{marraythisX[12] =
dubthisslice[(j+(3*m+3))];} catch (Exception e) {
        try{marraythisX[12] =
dubthisslice[(j+(2*m+2))];} catch (Exception ee) {

        try{marraythisX[12]=dubthisslice[j+(m+1)];} catch (Exception
eee) {marraythisX[12]=dubthisslice[j];}}

        try{marray3P[0] = dubbeforeslice[j];
} catch (Exception e) {marray3P[0]=dubthisslice[j];}
        try{marray3P[1] = dubthisslice[(j-
m)];} catch (Exception e) {marray3P[1]=dubthisslice[j];}
        try{marray3P[2] = dubthisslice[(j-
1)];} catch (Exception e) {marray3P[2]=dubthisslice[j];}
        try{marray3P[3] = dubthisslice[j];
} catch (Exception e) {marray3P[3]=dubthisslice[j];}
        try{marray3P[4] =
dubthisslice[(j+1)];} catch (Exception
e) {marray3P[4]=dubthisslice[j];}
        try{marray3P[5] =
dubthisslice[(j+m)];} catch (Exception
e) {marray3P[5]=dubthisslice[j];}
        try{marray3P[6] = dubafterslice[j];
} catch (Exception e) {marray3P[6]=dubthisslice[j];}

        try{marray3Xa[0] = dubbeforeslice[(j-
(m+1))];} catch (Exception e) {marray3Xa[0]=dubthisslice[j];}
        try{marray3Xa[1] = dubafterslice[(j+(m+1))];
} catch (Exception e) {marray3Xa[1]=dubthisslice[j];}
        try{marray3Xa[2] = dubthisslice[j];
} catch (Exception e) {marray3Xa[2]=dubthisslice[j];}
        try{marray3Xa[3] = dubbeforeslice[(j+(m-
1))];} catch (Exception e) {marray3Xa[3]=dubthisslice[j];}
        try{marray3Xa[4] = dubafterslice[(j-(m-1))];
} catch (Exception e) {marray3Xa[4]=dubthisslice[j];}

        try{marray3Xb[0] = dubbeforeslice[(j-
m)];} catch (Exception e) {marray3Xb[0]=dubthisslice[j];}
        try{marray3Xb[1] = dubafterslice[(j+m)];
} catch (Exception e) {marray3Xb[1]=dubthisslice[j];}
        try{marray3Xb[2] = dubthisslice[j];
} catch (Exception e) {marray3Xb[2]=dubthisslice[j];}
        try{marray3Xb[3] =
dubbeforeslice[(j+m)];} catch (Exception
e) {marray3Xb[3]=dubthisslice[j];}
        try{marray3Xb[4] = dubafterslice[(j-m)];
} catch (Exception e) {marray3Xb[4]=dubthisslice[j];}

        try{marray3Xc[0] = dubbeforeslice[(j-(m-
1))];} catch (Exception e) {marray3Xc[0]=dubthisslice[j];}
        try{marray3Xc[1] = dubafterslice[(j+(m-1))];
} catch (Exception e) {marray3Xc[1]=dubthisslice[j];}
        try{marray3Xc[2] = dubthisslice[j];
} catch (Exception e) {marray3Xc[2]=dubthisslice[j];}
        try{marray3Xc[3] = dubbeforeslice[(j+(m-
1))];} catch (Exception e) {marray3Xc[3]=dubthisslice[j];}

```

```

        try{marray3Xc[4] = dubafterslice[(j-(m-1))];
    }catch(Exception e){marray3Xc[4]=dubthisslice[j];}

        try{marray3Xd[0] = dubbeforeslice[(j-
1)];}catch(Exception e){marray3Xd[0]=dubthisslice[j];}
        try{marray3Xd[1] = dubafterslice[(j+1)];
    }catch(Exception e){marray3Xd[1]=dubthisslice[j];}
        try{marray3Xd[2] = dubthisslice[j];
    }catch(Exception e){marray3Xd[2]=dubthisslice[j];}
        try{marray3Xd[3] =
dubbeforeslice[(j+1)];}catch(Exception
e){marray3Xd[3]=dubthisslice[j];}
        try{marray3Xd[4] = dubafterslice[(j-1)];
    }catch(Exception e){marray3Xd[4]=dubthisslice[j];}

        medianarray[0] = median(marraythisX);
        medianarray[1] = median(marraythisP);
        medianarray[2] = median(marray3P);
        medianarray[3] = median(marray3Xa);
        medianarray[4] = median(marray3Xb);
        medianarray[5] = median(marray3Xc);
        medianarray[6] = median(marray3Xd);

        if(include)medianarray[7]= dubthisslice[j];
        filteredslice[j] = median(medianarray);

    }
    for(int
h=0;h<dimension;++h)dubthisslice[h]=filteredslice[h];
    }
}

prognum = (double) (i-1)/(stacksize);
progstr = nf.format(100*prognum);
IJ.showStatus("Filtro da Mediana - Estado do processo:
"+progstr+"% executado.");

if(atebit){
    newslice8 = double2byte(filteredslice);
    ip2 = imp2.getProcessor();
    ip2.setPixels(newslice8);
}
else{
    newslice = double2short(filteredslice);
    ip2 = imp2.getProcessor();
    ip2.setPixels(newslice);
}
stack2.addSlice(String.valueOf(i), ip2);

if(win!=null && !win.running){
    IJ.beep();
    quit=true;
    IJ.showMessage(new String("Processamento
interrompido!!!.\nNenhum resultado será gerado."));
    return imp2;
}

```

```

    }

    titlestring = "Janela "+sizes[(int)nsize]+" Filtro da
Mediana com x"+String.valueOf((int)times)+" repetições - Arquivo
original: "+otitle;
    imp2.setStack(titlestring, stack2);
    return imp2;
}

private static double median(double array[]){
    Arrays.sort(array);
    int len = array.length;
    if(len%2==0) return (array[(len/2)-1]+array[len/2])/2;
    else return array[(len-1)/2];
}

private short[] double2short(double array[]){
    short shortarray[] = new short[array.length];
    for(int j=0;j<array.length;++j){
        shortarray[j] = (short)array[j];
    }
    return shortarray;
}

private double[] short2double(short array[]){
    double doublearray[] = new double[array.length];
    for(int j=0;j<array.length;++j){
        doublearray[j] = (double)(0xffff & array[j]);
    }
    return doublearray;
}

private byte[] double2byte(double array[]){
    byte bytearray[] = new byte[array.length];
    for(int j=0;j<array.length;++j){
        bytearray[j] = (byte)array[j];
    }
    return bytearray;
}

private double[] byte2double(byte array[]){
    double doublearray[] = new double[array.length];
    for(int j=0;j<array.length;++j){
        doublearray[j] = (double)(0xff & array[j]);
    }
    return doublearray;
}
}

```

H.4 – Filtro da Gaussiana Ponderada (3D)

```

import ij.plugin.filter.ExtendedPlugInFilter;
import ij.plugin.filter.PlugInFilterRunner;
import ij.*;
import ij.gui.GenericDialog;

```

```

import ij.gui.DialogListener;
import ij.process.*;
import ij.plugin.ContrastEnhancer;
import java.awt.*;
import java.awt.event.*;
import ij.ImageStack;

/*****
* Bloco adicional de Filtragem (3D)
* Algoritmo utilizando o Filtro da Gaussiana Ponderada
* Ref.: Baseado no algoritmo do Tony Collins (2007)
* Características do algoritmo:
* 1) Permite seleccionar o tamanho da janela (radio da janela).
* 2) Permite seleccionar o valor sigma (testados: 0.5 até 3.0).
* 3) Trabalha no filtrado de imagens 3D (stack).
*
* @author Edward Flórez Pacheco
* @versão 2.1
* @data 04/nov/2011
* Escola Politécnica da Universidade de São Paulo (EPUSP) - Brasil
* Laboratório de Engenharia Biomédica (LEB)
*****/

public class Sigma_Filter_3D implements ExtendedPlugInFilter,
DialogListener {

    private static double radius = 2.;
    private static double sigmaWidth = 2.;
    private static double minPixFraction = 0.2;
    private static boolean outlierAware = true;

    int flags =
DOES_ALL|SUPPORTS_MASKING|CONVERT_TO_FLOAT|SNAPSHOT|KEEP_PREVIEW|
PARALLELIZE_STACKS;
    private int nPasses = 1;
    private int pass;
    protected int kRadius; //Tamanho é (2*kRadius+1)^2
    protected int kNPoints;
    protected int[] lineRadius;

    public int setup(String arg, ImagePlus imp) {
        return flags;
    }

    public int showDialog(ImagePlus imp, String command,
PlugInFilterRunner pfr) {
        GenericDialog gd = new GenericDialog(command+"...");
        gd.addNumericField("Radio", radius, 1, 6, "Pixels");
        gd.addNumericField("Use Pixels dentro de", sigmaWidth, 1,
6, "Sigmas");
        gd.addNumericField("Fracção Mínima de pixels",
minPixFraction, 1);
        gd.addCheckbox("Cosciente atípico", outlierAware);

```

```

        gd.addPreviewCheckbox(pfr);
gd.addDialogListener(this);
        gd.showDialog();
        if (gd.wasCanceled()) return DONE;
        IJ.register(this.getClass());
        return IJ.setupDialog(imp, flags);
    }

    public boolean dialogItemChanged(GenericDialog gd, AWTEvent e)
    {
        radius = gd.getNextNumber();
        sigmaWidth = gd.getNextNumber();
        minPixFraction = gd.getNextNumber();
        outlierAware = gd.getNextBoolean();
        if (gd.invalidNumber() || radius<0.5 || radius>11 ||
minPixFraction<0. || minPixFraction>1.)
            return false;
        makeKernel(radius);
        return true;
    }

    public void run(ImageProcessor ip) {

        int[] lineRadius;
        int kRadius, kNPoints, minPixNumber;
        synchronized(this) {
lineRadius = (int[]) (this.lineRadius.clone());
            kRadius = this.kRadius;
            kNPoints = this.kNPoints;
            minPixNumber = (int) (kNPoints * minPixFraction +
0.999999);
        }
        if (Thread.currentThread().isInterrupted()) return;
        pass++;
        doFiltering((FloatProcessor)ip, kRadius, lineRadius,
sigmaWidth, minPixNumber, outlierAware);
    }

    public void doFiltering(FloatProcessor ip, int kRadius, int[]
lineRadius, double sigmaWidth, int minPixNumber, boolean
outlierAware) {

        float[] pixels = (float[])ip.getPixels();
        int width = ip.getWidth();
        int height = ip.getHeight();
        Rectangle roi = ip.getRoi();
        int xmin = roi.x - kRadius;
        int xEnd = roi.x + roi.width;
        int xmax = xEnd + kRadius;
        int kSize = 2*kRadius + 1;
        int cacheWidth = xmax - xmin;
        int xminInside = xmin>0 ? xmin : 0;
        int xmaxInside = xmax<width ? xmax : width;
        int widthInside = xmaxInside - xminInside;
        boolean smallKernel = kRadius < 2;
        float[] cache = new float[cacheWidth*kSize];

```

```

        for (int y=roi.y-kRadius, iCache=0; y<roi.y+kRadius; y++)
            for (int x=xmin; x<xmax; x++, iCache++)
cache[iCache] = pixels[(x<0 ? 0 : x>=width ? width-1 : x) +
width*(y<0 ? 0 : y>=height ? height-1 : y)];
        int nextLineInCache = 2*kRadius;
        double[] sums = new double[2];
        Thread thread = Thread.currentThread();
        long lastTime = System.currentTimeMillis();
        for (int y=roi.y; y<roi.y+roi.height; y++) {
            long time = System.currentTimeMillis();
            if (time-lastTime > 100) {
                lastTime = time;
                if (thread.isInterrupted()) return;
                showProgress(y/(double)(roi.height));
            }
            int ynext = y+kRadius;
            if (ynext >= height) ynext = height-1;
            float leftpxl = pixels[width*ynext];
            float rightpxl = pixels[width-1+width*ynext];
            int iCache = cacheWidth*nextLineInCache;
            for (int x=xmin; x<0; x++, iCache++)
                cache[iCache] = leftpxl;
            System.arraycopy(pixels, xminInside+width*ynext,
cache, iCache, widthInside);
            iCache += widthInside;
            for (int x=width; x<xmax; x++, iCache++)
                cache[iCache] = rightpxl;
            nextLineInCache = (nextLineInCache + 1) % kSize;
            boolean fullCalculation = true;

            for (int x=roi.x, p=x*y*width, xCache0=kRadius;
x<xEnd; x++, p++, xCache0++) {
                double value = pixels[p];

                if (fullCalculation) {
                    fullCalculation = smallKernel;
                    getAreaSums(cache, cacheWidth, xCache0,
lineRadius, kSize, sums);
                } else
                    addSideSums(cache, cacheWidth, xCache0,
lineRadius, kSize, sums);
                double mean = sums[0]/kNPoints;
                double variance = sums[1]/kNPoints - mean*mean;
                double sigmaRange =
sigmaWidth*Math.sqrt(variance);
                double sigmaBottom = value - sigmaRange;
                double sigmaTop = value + sigmaRange;
                double sum = 0;
                int count = 0;
                for (int y1=0; y1<kSize; y1++) {
for (int x1=xCache0-lineRadius[y1], iCache1=y1*cacheWidth+x1;
x1<=xCache0+lineRadius[y1]; x1++, iCache1++) {
                    float v = cache[iCache1];
                    if ((v>=sigmaBottom)&&(v<=sigmaTop)) {
                        sum += v;
                        count++;
                    }
                }
            }
        }
    }

```



```

        }
    }
}

    if (count >= minPixNumber)
        pixels[p] = (float)(sum/count);
    else {
        if (outlierAware)
            pixels[p] = (float)((sums[0]-
value)/(kNPoints-1));
        else
            pixels[p] = (float)mean;
    }
}
int newLineRadius0 = lineRadius[kSize-1];
System.arraycopy(lineRadius, 0, lineRadius, 1, kSize-
1);
lineRadius[0] = newLineRadius0;
}
}

private void getAreaSums(float[] cache, int cacheWidth, int
xCache0, int[] lineRadius, int kSize, double[] sums) {
    double sum=0, sum2=0;
    for (int y=0; y<kSize; y++) {
        for (int x=xCache0-lineRadius[y],
iCache=y*cacheWidth+x; x<=xCache0+lineRadius[y]; x++, iCache++) {
            float v = cache[iCache];
            sum += v;
            sum2 += v*v;
        }
    }
    sums[0] = sum;
    sums[1] = sum2;
    return;
}

private void addSideSums(float[] cache, int cacheWidth, int
xCache0, int[] lineRadius, int kSize, double[] sums) {
    double sum=0, sum2=0;
    for (int y=0; y<kSize; y++) {
        int iCache0 = y*cacheWidth + xCache0;
        float v = cache[iCache0 + lineRadius[y]];
        sum += v;
        sum2 += v*v;
        v = cache[iCache0 - lineRadius[y] - 1];
        sum -= v;
        sum2 -= v*v;
    }
    sums[0] += sum;
    sums[1] += sum2;
    return;
}

```

```

public synchronized void makeKernel(double radius) {
    if (radius>=1.5 && radius<1.75)
        radius = 1.75;
    else if (radius>=2.5 && radius<2.85)
        radius = 2.85;
    int r2 = (int) (radius*radius) + 1;
    kRadius = (int) (Math.sqrt(r2+1e-10));
    lineRadius = new int[2*kRadius+1];
    lineRadius[kRadius] = kRadius;
    kNPoints = 2*kRadius + 1;
    for (int y=1; y<=kRadius; y++) {
        int dx = (int) (Math.sqrt(r2-y*y+1e-10));
        lineRadius[kRadius+y] = dx;
        lineRadius[kRadius-y] = dx;
        kNPoints += 4*dx + 2;
    }
}

public void setNPASSES (int nPasses) {
    this.nPasses = nPasses;
    pass = 0;
}

private void showProgress(double percent) {
    percent = (double) (pass-1)/nPasses + percent/nPasses;
    IJ.showProgress(percent);
}
}

```

H.5 – Filtro Anscombe/Wiener (3D)

```

import ij.*;
import ij.process.*;
import ij.plugin.filter.*;
import ij.gui.*;
import java.text.NumberFormat;
import ij.ImageStack;

/*****
* Bloco de Filtragem (3D)
* Algoritmo para fazer a filtragens das imagens 3D reais e/ou
* simuladas
* Ref.: Restauração de imagens digitais utilizando o Filtro de
* Wiener (Lee) no dominio de Anscombe
* e o filtro inverso de Anscombe no dominio da frequencia, 2009
* Romualdo, L.C.S.
* Caracteristicas do algoritmo:
* 1) Uma vez introduzido o ruído Poisson na imagem 3D (real ou
* simulada)
* 2) Aplicamos a Transformada de Anscombe (AT) pra converter o
* ruído Poisson em ruído Gaussiano

```

```

* 3) É aplicado o Filtro de Wiener (Lee)
* 4) Finalmente, a imagem 3D filtrada no passo (3) é submetida à
* Transformada Inversa de Anscombe (IAT)
* 5) Obtem-se a imagem filtrada.
*
* @author Edward Flórez Pacheco
* @versão 3.2
* @data 03/mar/2011
* Escola Politecnica da Universidade de São Paulo (EPUSP) - Brazil
* Laboratório de Engenharia Biomédica (LEB)
*****/

```

```

public class Filtro_adaptativo_Wiener_TAnscombe_3D_v2 implements
PlugInFilter {

```

```

    private ImagePlus imp;
    private ImagePlus imp2;
    private ImageStack stack;
    private ImageStack stack2;
    private ImageProcessor ip;
    //private ImageProcessor ip2;
    private boolean atebit = false;
    private boolean include = false;
    private NumberFormat nf = NumberFormat.getInstance();
    //private double prognum;
    //private String progstr;
    private String otitle;
    //private String titlestring;
    private ImageWindow win;
    private boolean quit = false;
    private int s = 0;
    private int N=15;
    private int nc=N/2;
    private int contador=0;

    public int setup(String argv, ImagePlus imp){
        try{stack = imp.getStack();}
        catch(Exception e){
            IJ.showMessage("Deve-se ter uma stack aberta!!");
            return DONE;
        }
        otitle = imp.getTitle();
        GenericDialog d = new GenericDialog("Filtro Adaptativo
Wiener/Anscombe", IJ.getInstance());
        d.addCheckbox("Iniciando a filtragem...",include);
        d.showDialog();
        if(d.wasCanceled())return DONE;
        include = d.getNextBoolean();
        nf.setMaximumFractionDigits(2);
        nf.setMinimumFractionDigits(2);
        this.imp = imp;
        if(imp.getBitDepth()==8)atebit=true;
        win = imp.getWindow();
        if (win!=null) win.running = true; //wsr
        return DOES_8G+DOES_16+STACK_REQUIRED+DOES_STACKS+NO_UNDO;
    }
}

```

```

    }
    public void run(ImageProcessor ip){

        int m = stack.getWidth();
        int n = stack.getHeight();
        imp2 = new ImagePlus("Filtro Anscombe/Wiener",
imp.getStack());
        stack2 = imp2.createEmptyStack();
        int stacksize = stack.getSize();

        // transformando em float p/ ficar independente do tipo
        FloatProcessor ip_float =
(FloatProcessor)ip.convertToFloat();
        ImagePlus imp_float = new ImagePlus("Float", ip_float);
        FloatProcessor ip_float2 =
(FloatProcessor)ip.convertToFloat();
        ImagePlus imp_float2 = new ImagePlus("Float", ip_float2);

        //float [][] matA = new
float[stack.getWidth()][stack.getHeight()];
        float [][] matb = new
float[stack.getWidth()][stack.getHeight()];
        float [][] matLee = new
float[stack.getWidth()][stack.getHeight()];

        //////////////////////////////////////
        //Passo N°2, Transformada de Anscombe (AT)
        //////////////////////////////////////

        for(s = 0; s<=stacksize; s++){

            for(int i = 0;i<stack.getWidth();i++){
                for(int j = 0 ;j<stack.getHeight();j++){
                    int y =(int) ip_float.getf(i,j);
                    matb[i][j]=(int) (2*Math.sqrt(y+(3/8)));
                }
            }

        }

        //////////////////////////////////////
        //Passo N°3, Filtro de Wiener (Lee)
        //////////////////////////////////////

        for(int y=nc; y< n-nc; y++) { // nao considera
bordas

            for (int x=nc; x<m-nc; x++){
                float mediaLocal=0;
                contador++;
                int contadorMatriz=0;
                for(int r=-nc; r<=nc; r++){
                    for(int c=-nc; c<=nc; c++){
                        contadorMatriz++;

```

```

mediaLocal=(int)
(mediaLocal+matb[x-c][y-r]);
    }
    mediaLocal=mediaLocal/contadorMatriz;
    float desviacionMatriz=0;
    int somal=0;
    for(int r=-nc; r<=nc; r++)
        for(int c=-nc; c<=nc; c++){
            somal=(int) (somal+Math.pow((matb[x-
c][y-r]-mediaLocal), 2));
        }
    desviacionMatriz=(float)
Math.sqrt(somal/(contadorMatriz-1));
    if(desviacionMatriz==0)desviacionMatriz=1;
    float K= (1-
((1)/(desviacionMatriz*desviacionMatriz)));
    if(K<0) K=0;
    if(K>1) K=1;
    float d=(K*matb[x][y])+((1-K)*mediaLocal);
    matLee[x][y]=d;
    }
}

////////////////////////////////////
//Passo N°4, Tranformada Inversa de Anscombe (IAT)
////////////////////////////////////

    for(int i = 0;i<stack.getWidth();i++){
        for(int j = 0 ;j<stack.getHeight();j++){
            float r=((matLee[i][j]*matLee[i][j])/4)-
(1/8);
            ip_float2.setf(i, j, r);
        }
    }
    imp_float2.show();
}
}
}

```

H.6 – Segmentação de Imagens (3D)

```

import java.util.LinkedList;
import java.util.Vector;
import ij.IJ;
import ij.ImageStack;

/*****
* Conjunto de métodos de segmentação.
* Utiliza o conceito Fuzzy Connectedness
* segmenta imagens por varios métodos:
* - Genérico, LIFO, kTetaFOEMS, Pesos Dinâmicos.
*
* @author S.Furuie / Edward Flórez Pacheco
* @versão 4.2
* @data 03/09/2010
* Escola Politecnica da Universidade de São Paulo - Brazil

```

```

* Laboratório de Engenharia Biomédica (LEB)
*****/

public class Segmentador {
    private int nVol, nSli, nLin, nCol;
    private float[] f_in;
    private double fp_min, fp_max;

    Segmentador(ImageStack stack, float[] f_in){
        this.f_in=f_in;
        nVol=1;
        nSli=stack.getSize(); nLin=stack.getHeight();
        nCol=stack.getWidth();

        fp_min=fp_max= f_in[0];
        for(int i=0; i<f_in.length; i++) {
            double temp=f_in[i];
            if(temp > fp_max) fp_max=temp;
            else if(temp < fp_min) fp_min=temp;
        }
    }

    // ===== métodos =====

    public float[] kFOEMS(Vector<Spel> sementes, double lambda,
        boolean IJresults_presente){
        int nSpels=nVol*nSli*nLin*nCol;
        final int Q_capacidadeInicial = nLin*nCol;
        final double epsilon=1.0e-4;
        final int MAX_ADJACENTES=50;

        //Inicializando
        float[] fconn=new float[nSpels];

        double sigma_min=(fp_max-fp_min)/100.0;
        if(IJresults_presente)
            IJ.write("Imagem:fp_max= "+IJ.d2s(fp_max)+" fp_min=
"+IJ.d2s(fp_min)+" com "+nSpels+" spels");
            // 1) Inicializando o fuzzy set fconn com 0 e 1(sementes)
            inicializaFuzzySet(fconn, sementes); // zera todos, exceto 1
            na posicao sementes

        // Obtendo as características das sementes
        double[] m1s1m2s2=obter_m1s1_m2s2_2D(sementes, sigma_min,
        IJresults_presente);
        if(IJresults_presente){
            IJ.write("Final: \n (m1,s1)=( " + IJ.d2s(m1s1m2s2[0])+ ", "+
            IJ.d2s(m1s1m2s2[1])+") \n (m2,s2)=( " +
            IJ.d2s(m1s1m2s2[2])+", "+IJ.d2s(m1s1m2s2[3])+")");
            IJ.write("Vetor de sementes com
"+sementes.size()+"elementos");
        }
        // Aplicando o algoritmo .....
        // Iniciando o Queue e marcando se spel presente no queue
        // 2) push all spels c E C such that, for some s E C, uk(s,c)>0 to
        Queue (c é adjacente a s)

```

```

Vector<Spel> Queue=new Vector<Spel>(Q_capacidadeInicial);
byte[] QueuePresenca=new byte[nSpels]; for(int i=0;
i<nSpels; i++)QueuePresenca[i]=0;
Vector<Spel> cAdjacentes;
for(int i=0; i<sementes.size(); i++) // obtendo os vizinhos
nao sementes de cada semente

{cAdjacentes=sementes.elementAt(i).getAdjacentesNdNaoSementes(fconn);
for(int a=0; a< cAdjacentes.size(); a++){ // p/ cada
vizinho, verificar se ja' nao esta' no Queue
Spel d=cAdjacentes.elementAt(a);
int d_index=d.getIndex();
if(QueuePresenca[d_index]==0){// nao precisa testar
afinidade pois com certeza  $uk(s,c) > 0$ [exponencial]
Queue.add(d);
QueuePresenca[d_index]=1;
}
}
}
if(IJresults_presente){
IJ.write("Queue de spels nao-sementes="+Queue.size());
}

/**
 * 3) While Q is not empty do
 * 4) remove a spel c from Q
 * 5) find flmax=max(min(f1(d),uk(c,d)) d E C (d é
 * adjacente a c)
 */

int contadorPops=0;
float fld, flmax, temp;
double[] uk=new double[MAX_ADJACENTES];
while (!Queue.isEmpty()){
if((contadorPops % 1000000) ==0){
if(IJresults_presente)IJ.write("Pop num:
"+contadorPops+" com Q size atual="+Queue.size());
}
contadorPops++;
Spel c=(Spel)Queue.remove(0);
int c_index=c.getIndex();
QueuePresenca[c_index]=0;
cAdjacentes=c.getAdjacentesNd();
flmax=(float)0.0; int i_flmax=0;
for(int i=0; i< cAdjacentes.size(); i++){
Spel d=cAdjacentes.elementAt(i);
int d_index=d.getIndex();
fld=fconn[d_index];
if (fld > epsilon)

{uk[i]=c.getAfinidadeComWeight(f_in[c_index],f_in[d_index],
m1s1m2s2[0], m1s1m2s2[1], m1s1m2s2[2],
m1s1m2s2[3], 0.8);

temp=(float) Math.min(uk[i], fld);

```

```

        if(temp > flmax){flmax=temp;
            i_flmax=i;
        }
    }
}

/**
 * 6) if flmax > f1(c) then
 * 7) set f1(c)=flmax;
 * 8) push all spels e such that
 * min[flmax, uk(c,e)] > f1(e) to Q endif endwhile
 */

if((flmax > fconn[c_index]) && (flmax > epsilon)){
    fconn[c_index]=flmax;
    for(int i=0; i< cAdjacentes.size(); i++)

        {if(i==i_flmax) continue;
        Spel d=cAdjacentes.elementAt(i);
        int d_index=d.getIndex();
        fld=fconn[d_index]; //f1.getf(d_index);
        if (fld < (1-epsilon))
            {
                if(Math.min(uk[i], flmax) > (fld +
epsilon))){
                    if (QueuePresenca[d_index]==0 ){
                        Queue.add(d);
                        QueuePresenca[d_index]=1;
                    }
                }
            }
        }
    }
}

return(fconn);
}

// ===== métodos =====
public float[] kFOEMS_DynamicWeight(Vector<Spel> sementes,
    boolean IJresults_presente){
    int nSpels=nVol*nSli*nLin*nCol;
    final int Q_capacidadeInicial = nLin*nCol;
    final double epsilon=1.0e-4;
    final int MAX_ADJACENTES=50;

    // Inicializando .....
    float[] fconn=new float[nSpels];
    double sigma_min=(fp_max-fp_min)/100.0;
    if(IJresults_presente)
        IJ.write("fp_max= "+IJ.d2s(fp_max)+" fp_min=
"+IJ.d2s(fp_min)+" com "+nSpels+" spels");
    // 1) Inicializando o fuzzy set f1 com 0 e 1(sementes)
    inicializaFuzzySet(fconn, sementes); // zera todos
    // Obtendo as características das sementes

```



```

    double[] m1s1m2s2=obter_m1s1_m2s2_2D(sementes, sigma_min,
IJresults_presente);
    if(IJresults_presente){
        IJ.write("Final: \n (m1,s1)=( " + IJ.d2s(m1s1m2s2[0])+ ", "+
            IJ.d2s(m1s1m2s2[1])+") \n (m2,s2)=( " +
IJ.d2s(m1s1m2s2[2])+", "+IJ.d2s(m1s1m2s2[3])+") ");
        IJ.write("Vetor de sementes com
"+sementes.size()+"elementos");
    }
// Aplicando o algoritmo .....
// Iniciando o Queue e marcando se spel presente no queue
// 2) push all spels c E C such that, for some s E C, uk(s,c)>0 to
Queue (c
// é adjacente a s)
    Vector<Spel> Queue=new Vector<Spel>(Q_capacidadeInicial);
    byte[] QueuePresenca=new byte[nSpels]; for(int i=0;
i<nSpels; i++)QueuePresenca[i]=0;
    Vector<Spel> cAdjacentes;
    for(int i=0; i<sementes.size(); i++)

{cAdjacentes=sementes.elementAt(i).getAdjacentesNdNaoSementes(fcon
n);
        for(int a=0; a< cAdjacentes.size(); a++){
            Spel d=cAdjacentes.elementAt(a);
            int d_index=d.getIndex();
            if(QueuePresenca[d_index]==0){
                Queue.add(d);
                QueuePresenca[d_index]=1;
            }
        }
    }
    if(IJresults_presente){
        IJ.write("Queue de spels nao-sementes="+Queue.size());
    }
    /**
    * 3) While Q is not empty do
    * 4) remove a spel c from Q
    * 5) find flmax=max(min(fl(d),uk(c,d)) d E C (d é
adjacente a c)
    */
    int contadorPops=0;
    double fld, temp, flmax;
    double[] uk=new double[MAX_ADJACENTES];
    while (!Queue.isEmpty()){
        if((contadorPops % 1000000) ==0){
            if(IJresults_presente)IJ.write("Pop num:
"+contadorPops+" com Q size atual="+Queue.size());
        }
        contadorPops++;
        Spel c=(Spel)Queue.remove(0);
        QueuePresenca[c.getIndex()]=0;
        cAdjacentes=c.getAdjacentesNd();
        flmax=0.0; int i_flmax=0;
        for(int i=0; i< cAdjacentes.size(); i++)
            {Spel d=cAdjacentes.elementAt(i);

```

```

        fld=fconn[d.getIndex()];
        if (fld > epsilon)

{uk[i]=c.getAfinidadeDynamicWeight(f_in[c.getIndex()],f_in[d.getIndex()],
                                     m1s1m2s2[0], m1s1m2s2[1], m1s1m2s2[2],
m1s1m2s2[3]);

    // ATTENTION pode store uk p/ agilizar
    temp=Math.min(uk[i], fld);
    if(temp > flmax){flmax=temp;
                    i_flmax=i;
                }
        }
    }

/**
 * 6) if flmax > f1(c) then
 * 7) set f1(c)=flmax;
 * 8) push all spels e such that
 * min[flmax, uk(c,e)] > f1(e) to Q endif endwhile
 */
    int c_index=c.getIndex();
    if((flmax > fconn[c_index]) && (flmax > epsilon)){
//forçar recalculo dos vizinhos pois pode ter melhorado
        fconn[c_index]=(float)flmax;
        for(int i=0; i< cAdjacentes.size(); i++)
            {if(i==i_flmax) continue;
              Spel d=cAdjacentes.elementAt(i);
              int d_index=d.getIndex();
              fld=fconn[d_index]; //f1.getf(d_index);
              if (fld < (1-epsilon))
                  {

                        if(Math.min(uk[i], flmax) > (fld +
epsilon))){
                                if (QueuePresenca[d_index]==0 ){
Queue.add(d);
                                QueuePresenca[d_index]=1;
                                }
                            }
                        }
                    }
                }
        }
    }
return(fconn);
}

// ===== métodos =====
/**
 * segmenta somente na regio na qual mask()!=0
 */
    public float[] kFOEMS_MASK(Vector<Spel> sementes, byte[] mask,
double lambda,
                                boolean IJresults_presente){
    int nSpels=nVol*nSli*nLin*nCol;

```

```

final int Q_capacidadeInicial = nLin*nCol;
final double epsilon=1.0e-4;
final int MAX_ADJACENTES=50;

// Inicializado
float[] fconn=new float[nSpels];
double sigma_min=(fp_max-fp_min)/100.0;
if(IJresults_presente)
    IJ.write("fp_max= "+IJ.d2s(fp_max)+" fp_min=
"+IJ.d2s(fp_min)+" com "+nSpels+" spels");
    // 1) Inicializando o fuzzy set f1 com 0 e 1(sementes)
    inicializaFuzzySet(fconn, sementes); // zera todos

    // Obtendo as caracteristicas das sementes
    double[] m1s1m2s2=obter_m1s1_m2s2_2D(sementes, sigma_min,
IJresults_presente);
    if(IJresults_presente){
        IJ.write("Final: \n (m1,s1)=( " + IJ.d2s(m1s1m2s2[0])+ ", "+
IJ.d2s(m1s1m2s2[1])+") \n (m2,s2)=( " +
IJ.d2s(m1s1m2s2[2])+", "+IJ.d2s(m1s1m2s2[3])+")");
        IJ.write("Vetor de sementes com
"+sementes.size()+"elementos");
    }
// Aplicando o algoritmo
// Iniciando o Queue e marcando se spel presente no queue
// 2) push all spels c E C such that, for some s E C, uk(s,c)>0 to
Queue (c
// é adjacente a s)
Vector<Spel> Queue=new Vector<Spel>(Q_capacidadeInicial);
byte[] QueuePresenca=new byte[nSpels]; for(int i=0;
i<nSpels; i++)QueuePresenca[i]=0;
Vector<Spel> cAdjacentes;
for(int i=0; i<sementes.size(); i++)

{cAdjacentes=sementes.elementAt(i).getAdjacentesNdNaoSementes(fcon
n);
    for(int a=0; a< cAdjacentes.size(); a++){
        Spel d=cAdjacentes.elementAt(a);
        int d_index=d.getIndex();
        if(mask[d_index] !=0) {
            if(QueuePresenca[d_index]==0) {
                Queue.add(d);
                QueuePresenca[d_index]=1;
            }
        }
    }
}
if(IJresults_presente){
    IJ.write("Queue de spels nao-sementes="+Queue.size());
}
/**
 * 3) While Q is not empty do
 * 4) remove a spel c from Q
 * 5) find flmax=max(min(f1(d),uk(c,d)) d E C (d é
adjacente a c)
 */

```

```

    int contadorPops=0;
    double fld, temp, flmax;
    double[] uk=new double[MAX_ADJACENTES];
    while (!Queue.isEmpty()){
        if((contadorPops % 1000000) ==0){
            if(IJresults_presente)IJ.write("Pop num:
"+contadorPops+" com Q size atual="+Queue.size());
        }
        contadorPops++;
        Spel c=(Spel)Queue.remove(0);

        int c_index=c.getIndex();
        QueuePresenca[c_index]=0;
        cAdjacentes=c.getAdjacentesNd();
        flmax=0.0; int i_flmax=0;
        for(int i=0; i< cAdjacentes.size(); i++)
            {Spel d=cAdjacentes.elementAt(i);
            int d_index=d.getIndex();
            if(mask[d_index] ==0) continue;
            fld=fconn[d_index];
            if (fld > epsilon)

            {uk[i]=c.getAfinidadePorValor(f_in[c_index],f_in[d_index],
                                            m1s1m2s2[0], m1s1m2s2[1], m1s1m2s2[2],
m1s1m2s2[3], lambda);

            // ATTENTION pode store uk p/ agilizar
            temp=Math.min(uk[i], fld);
            if(temp > flmax){flmax=temp;
                i_flmax=i;
            }
        }
    }

    /**
     * 6) if flmax > f1(c) then
     * 7) set f1(c)=flmax;
     * 8) push all spels e such that
     * min[flmax, uk(c,e)] > f1(e) to Q endif endwhile
     */
    if((flmax> fconn[c_index]) && (flmax > epsilon)){
        fconn[c_index]=(float)flmax;
        for(int i=0; i< cAdjacentes.size(); i++)
            {if(i==i_flmax) continue;
            Spel d=cAdjacentes.elementAt(i);
            int d_index=d.getIndex();
            if(mask[d_index] ==0) continue;
            fld=fconn[d_index];
            if (fld < (1-epsilon))
            {
                if(Math.min(uk[i], flmax) > (fld +
epsilon)){
                    if (QueuePresenca[d_index]==0 ){
                        Queue.add(d);
                        QueuePresenca[d_index]=1;
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
}
return(fconn);
}
public byte[] kTetaFOEMS(Vector<Spel> sementes, double teta,
double lambda,
boolean IJresults_presente){
    int nSpels=nVol*nSli*nLin*nCol;
    final int Q_capacidadeInicial = nLin*nCol;

    // Inicializando
    byte[] fconn=new byte[nSpels];

    double sigma_min=(fp_max-fp_min)/100.0;
    if(IJresults_presente)
        IJ.write("fp_max= "+IJ.d2s(fp_max)+" fp_min=
"+IJ.d2s(fp_min)+" com "+nSpels+" spels");
        // 1) Inicializando o fuzzy set f1 com 0 e 1(sementes)
        inicializaFuzzySet(fconn, sementes); // zera todos
        // Obtendo as caracteristicas das sementes
        double[] m1s1m2s2=obter_m1s1_m2s2_2D(sementes,
sigma_min, IJresults_presente);
        if(IJresults_presente){
            IJ.write("Final: \n (m1,s1)=( " + IJ.d2s(m1s1m2s2[0]))+
", "+
IJ.d2s(m1s1m2s2[1]))+" \n (m2,s2)=( " +
IJ.d2s(m1s1m2s2[2]))+" "+IJ.d2s(m1s1m2s2[3]))+" " );
            IJ.write("Vetor de sementes com
"+sementes.size()+"elementos");
        }
        // Aplicando o algoritmo .....
        // 2) push all spels c E C such that, for some s E C,
uk(s,c)>0 to Queue (c
// é adjacente a s)
        Vector<Spel> Queue=new
Vector<Spel>(Q_capacidadeInicial);
        for(int i=0; i<sementes.size(); i++) // obtendo os
vizinhos nao sementes de cada semente

Queue.addAll(sementes.elementAt(i).getAdjacentesNdNaoSementesTetaA
fim
(f_in,fconn,teta,m1s1m2s2[0],
m1s1m2s2[1], m1s1m2s2[2], m1s1m2s2[3], lambda));
        if(IJresults_presente){
            IJ.write("Queue de spels nao-
sementes="+Queue.size());
        }
        /**
        * 3) While Q is not empty do
        * 4) remove a spel c from Q
        * 5) if f1(c) != 1 {
        * 6) f1(c)=1
        * 7) push spels d=adjacentes(c) | uk(c,d)>= teta }

```

```

        */
        int contadorPops=0,c_index;
        double uk;
        Vector<Spel> cAdjacentes;
        while (!Queue.isEmpty()){
            if((contadorPops % 100000) ==0)
                if(IJresults_presente)IJ.write("Pop num:
"+contadorPops+" com Q size atual="+Queue.size());
            contadorPops++;
            Spel c=(Spel)Queue.remove(Queue.size()-1);
            c_index=c.getIndex();
            if(fconn[c_index]!=1){
                fconn[c_index]=1;
                cAdjacentes=c.getAdjacentesNd();
                for(int i=0; i< cAdjacentes.size(); i++){
                    Spel d=cAdjacentes.elementAt(i);

                    uk=c.getAfinidadePorValor(f_in[c_index],f_in[d.getIndex()],
                                                m1s1m2s2[0],
m1s1m2s2[1], m1s1m2s2[2], m1s1m2s2[3], lambda);
                    if(uk >= teta) Queue.add(d);
                }
            }
        }

        return(fconn);
    }

// ===== métodos =====
/**
 * P/ agilizar a localização do spel com maior afinidade, usa-se a
 * abordagem de Dial em vez de operar com um Queue tradicional:
 * vetor (Q[f]) onde f é inteiro e representa a afinidade entre
 * [0..MAX_AFIN]e Q[f] é o index do spel. Desse modo, o vetor é
 * ordenado (crescente) em termos de afinidade.
 * Cada posicao f é um bucket que pode ter 0, 1 ou varios spels.
 * Usaremos uma lista.
 * LIFO: true =>spel com mesma afinidade "last in" no bucket sera'
 * "first out" p/ incorporar vizinhos
 * Algoritmo
 * 1)Inicializar o fuzzy set fconn com 0 e MAX_AFIN(sementes)
 * 2)push sementes/spel s[i] to Q => push spel s[i] to bucket
 * Q[index], onde index=f(s)
 * 3) While Q is not empty do
 * 4) remove a spel c from Q for which f[c] is maximal
 * 5) for each spel e such that uk(c,e)>0 (adjacente) do
 * 6)     set fmin=min(f[c],uk(c,e))
 * 7)     if (fmin > f[e] then
 * 8)         //set f[e]=fmin
 * 9)         //if (e is already in Q) then remove e//update the
 * location of e in Q
 * 10)        //else push e to Q
 * 11)            set f[e]=fmin
 * 12)            push e to Q
 * OBS.:a) desempenho melhora bastante eliminando-se 8,9 e10. Em
 * vez de verificar e remover,

```

```

*           a ideia e' deixar p/ o loop externo.
*/

    public float[] Dijkstra_Dial(Vector<Spel> sementes, double
lambda,
                                boolean IJresults_presente, boolean LIFO){

    int nSpels=nVol*nSli*nLin*nCol;
    final int MAX_ADJACENTES=50;
    final int MAX_AFIN=100000;

    //Inicializando
    int[] fconn=new int[nSpels];
    float[] f_out=new float[nSpels];
    double sigma_min=(fp_max-fp_min)/100.0;

    if(IJresults_presente)
        IJ.write("Imagem:fp_max= "+IJ.d2s(fp_max)+" fp_min=
"+IJ.d2s(fp_min)+" com "+nSpels+" spels");
    // 1) Inicializando o fuzzy set fconn com 0 e MAX_AFIN(sementes)
        // Predecessor[i]=-1;
        // Label[i]=-1;
        inicializaFuzzySet(fconn, sementes, MAX_AFIN); // zera
        todos, exceto 1 na posicao sementes
        // Label[s_index]=s_index;
        // Obtendo as caracteristicas das sementes
        double[] m1s1m2s2=obter_m1s1_m2s2_2D(sementes, sigma_min,
IJresults_presente);
        if(IJresults_presente){
            IJ.write("Final: \n (m1,s1)=( " + IJ.d2s(m1s1m2s2[0])+
", "+
                        IJ.d2s(m1s1m2s2[1]))+" \n (m2,s2)=( " +
IJ.d2s(m1s1m2s2[2]))+" "+IJ.d2s(m1s1m2s2[3]))+"");
            IJ.write("Vetor de sementes com
"+sementes.size()+"elementos");
        }
    // Criando o vetor Q[f] com MAX_AFIN elementos, cada elemento
    uma lista LinkedList
        Vector<LinkedList<Spel>> Q=new
Vector<LinkedList<Spel>>(MAX_AFIN + 1);
        LinkedList<Spel> lk;
        for (int f=0; f<=MAX_AFIN; f++) {
            lk=new LinkedList<Spel>();
            Q.add(lk);
        }
    // 2) push sementes/spel s[i] to Q => push spel s[i] to bucket
    Q[index], onde index=fconn(s)
        for(int i=0; i<sementes.size(); i++)
        { int indexSpel=sementes.elementAt(i).getIndex();
          int indexQ=fconn[indexSpel];
          lk=Q.elementAt(indexQ); //
          lk.add(sementes.elementAt(i));
          Q.setElementAt(lk, indexQ);
        }
    int contadorPops=0;
    int fCorrente=MAX_AFIN;

```

```

    int flmin;
    double[] uk=new double[MAX_ADJACENTES];
    Spel c;
    Vector<Spel> cAdjacentes;
    /**
     * 3) While Q is not empty do
     * 4) remove a spel c from Q for which f[c] is maximal
     */
    while (fCorrente >0){
        if((contadorPops % 100000) ==0){
            if(IJresults_presente)IJ.write("Pop num:
"+contadorPops+" com ponteiro de custo f (deve ir a
0)="+fCorrente);
        }
        contadorPops++;
        c=(Spel)null;
        int f_c=0;
        for(int f=fCorrente; f>0 ; f--) {
            lk=Q.elementAt(f);
            if(lk.isEmpty()){
                fCorrente--;
                continue;
            }
            else {
                if(LIFO){
                    c=lk.removeLast();
                } else { // FIFO
                    c=lk.removeFirst();
                }
                f_c=f;
                break;
            }
        }
        if( f_c==0){fCorrente=0; break; }

    /** spel c com afinidade acum. f_c, ou seja fconn(c)=f_c
    * 5) for each spel e such that uk(c,e)>0 (adjacente) do
    * 6)     set fmin=min(f[c],uk(c,e))
    */

        int c_index=c.getIndex();
        cAdjacentes=c.getAdjacentesNd();
        for(int i=0; i< cAdjacentes.size(); i++)
            {Spel e=cAdjacentes.elementAt(i);
             int e_index=e.getIndex(); // obtendo uk(c,e)

            uk[i]=c.getAfinidadePorValor(f_in[c_index],f_in[e_index],
                                         m1s1m2s2[0], m1s1m2s2[1], m1s1m2s2[2],
            m1s1m2s2[3], lambda);

            // ATTENTION pode store uk p/ agilizar
            flmin=(int) (MAX_AFIN * uk[i]);
            if(f_c < flmin) flmin=f_c;

    /**
    * 7)     if (fmin > f[e] then
    * 8)         //set f[e]=fmin
    * 9)         //if (e is already in Q) then remove e//update the

```



```

* location of e in Q
* 10)                                     //else push e to Q
* 11)         set f[e]=fmin
* 12)         push e to Q
*/

        if(flmin > fconn[e_index]){
            fconn[e_index]=flmin;
            lk=Q.elementAt(fconn[e_index]);
            lk.add(e);
            Q.setElementAt(lk, fconn[e_index]);
        }
    }

    // convertendo p/ [0, 1]
    for(int i=0; i< nSpels; i++)
f_out[i]=(float)fconn[i]/MAX_AFIN;
return(f_out);
}

/** Versao com afinidades obtidas por tabela pre-calculada
* P/ agilizar a localizaçao do spel com maior afinidade, usa-se a
* abordagem de Dial
* em vez de operar com um Queue tradicional:
* vetor (Q[f]) onde f é inteiro e representa a afinidade entre
* [0..MAX_AFIN]e
* Q[f] é o index do spel. Desse modo, o vetor é ordenado
* (crescente) em termos de afinidade.
* Cada posicao f é um bucket que pode ter 0, 1 ou varios spels.
* Usaremos uma lista.
* LIFO: true =>spel com mesma afinidade "last in" no bucket sera'
* "first out" p/ incorporar vizinhos
* Algoritmo
* 1)Inicializar o fuzzy set fconn com 0 e MAX_AFIN(sementes)
* 2)push sementes/spel s[i] to Q => push spel s[i] to bucket
* Q[index], onde index=f(s)
* 3) While Q is not empty do
* 4) remove a spel c from Q for which f[c] is maximal
* 5) for each spel e such that uk(c,e)>0 (adjacente) do
* 6)     set fmin=min(f[c],uk(c,e))
* 7)     if (fmin > f[e] then
* 8)         //set f[e]=fmin
* 9)         //if (e is already in Q) then remove e//update the
* location of e in Q
* 10)                                     //else push e to Q
* 11)         set f[e]=fmin
* 12)         push e to Q
* OBS.:a) desempenho melhora bastante eliminando-se 8,9 e10. Em
* vez de verificar e remover,
* a ideia e' deixar p/ o loop externo.
*/

    public float[] Dijkstra_Dial_comTabelaAfin(Vector<Spel>
sementes, double lambda,

```

```

        boolean IJresults_presente, boolean LIFO){
int nSpels=nVol*nSli*nLin*nCol;
final int MAX_ADJACENTES=50;
final int MAX_AFIN=100000;    //

//Inicializando
int[] fconn=new int[nSpels];
float[] f_out=new float[nSpels];
double sigma_min=(fp_max-fp_min)/100.0;

    if(IJresults_presente)
        IJ.write("Imagem:fp_max= "+IJ.d2s(fp_max)+"   fp_min=
"+IJ.d2s(fp_min)+" com "+nSpels+" spels");
// 1) Inicializando o fuzzy set fconn com 0 e MAX_AFIN(sementes)
    // Predecessor[i]=-1;
    // Label[i]=-1;
    inicializaFuzzySet(fconn, sementes, MAX_AFIN); // zera
    todos, exceto 1 na posicao sementes
    // Label[s_index]=s_index;
    // Obtendo as caracteristicas das sementes
    double[] m1s1m2s2=obter_m1s1_m2s2_2D(sementes, sigma_min,
IJresults_presente);
    if(IJresults_presente){
        IJ.write("Final: \n (m1,s1)=( " + IJ.d2s(m1s1m2s2[0]))+
        ", "+
            IJ.d2s(m1s1m2s2[1]))+" \n (m2,s2)=( " +
IJ.d2s(m1s1m2s2[2]))+" "+IJ.d2s(m1s1m2s2[3]))+" " );
        IJ.write("Vetor de sementes com
"+sementes.size()+"elementos");
    }
    Vector<LinkedList<Spel>> Q=new
Vector<LinkedList<Spel>>(MAX_AFIN + 1);
    LinkedList<Spel> lk;
    for (int f=0; f<=MAX_AFIN; f++) {
        lk=new LinkedList<Spel>();
        Q.add(lk);
    }
    // Criando tabela de afinidades p/ agilizar obtencao de uk() e
    // evitar recalculos
    //tabelaAfin[ic*N256+id]=c.getAfinidadePorValor(f_in[c_index],f_in
    //[e_index],
    // m1s1m2s2[0], m1s1m2s2[1], m1s1m2s2[2], m1s1m2s2[3], lambda);
    // onde: fc=fmin+ic*(fmax-fmin)/255.0 e ic=round((fc-
    //fp_min)*255./(fp_max-fp_min))
    //      ic=round((fc-(float)fp_min)*temp2)

    int N256=256;
    float[] tabelaAfin=new float[N256*N256];
    float fc, fd, temp1, temp2;
    temp1=(float)((fp_max-fp_min)/255.0);
    temp2=(float)(255.0/(fp_max-fp_min));
    for (int ic=0; ic< N256; ic++){
        fc=ic*temp1 + (float)fp_min;
        for(int id=0; id<N256; id++){
            fd=id*temp1 + (float)fp_min;

```

```

        tabelaAfin[ic*N256 + id]=getAfinidadePorValor(fc,fd
,
        m1s1m2s2[0], m1s1m2s2[1], m1s1m2s2[2],
m1s1m2s2[3], lambda);
    }
}
// 2) push sementes/spel s[i] to Q => push spel s[i] to bucket
Q[index], onde index=fconn(s)
    for(int i=0; i<sementes.size(); i++) // para cada spel
semente
    { int indexSpel=sementes.elementAt(i).getIndex();
      int indexQ=fconn(indexSpel);
      lk=Q.elementAt(indexQ);
      lk.add(sementes.elementAt(i));
      Q.setElementAt(lk, indexQ);
    }
    int contadorPops=0;
    int fCorrente=MAX_AFIN;
    int flmin;
    double[] uk=new double[MAX_ADJACENTES];
    Spel c;
    Vector<Spel> cAdjacentes;

    /**
    * 3) While Q is not empty do
    * 4) remove a spel c from Q for which f[c] is maximal
    */
    while (fCorrente >0){ // fCorrente deve ir decrescendo a 0
        if((contadorPops % 1000000) ==0){
            if(IJresults_presente)IJ.write("Pop num:
"+contadorPops+" com ponteiro de custo f (deve ir a
0)="+fCorrente);
        }
        contadorPops++;
        c=(Spel)null;
        int f_c=0;
        for(int f=fCorrente; f>0 ; f--) {
            lk=Q.elementAt(f);
            if(lk.isEmpty()){
                fCorrente--;
                continue;
            }
            else { // Opcao LIFO
                if(LIFO){
                    c=lk.removeLast();

                } else { // FIFO
                    c=lk.removeFirst();
                }
                f_c=f;
                break;
            }
        }
        if( f_c==0){fCorrente=0; break; }

    /** spel c com afinidade acum. f_c, ou seja fconn(c)=f_c

```

```

* 5) for each spel e such that uk(c,e)>0 (adjacente) do
* 6)     set fmin=min(f[c],uk(c,e))
*/

    int c_index=c.getIndex();
    cAdjacentes=c.getAdjacentesNd();
    for(int i=0; i< cAdjacentes.size(); i++)
        {Spel e=cAdjacentes.elementAt(i);
         int e_index=e.getIndex();

        // ATTENTION pode store uk p/ agilizar
         int ic=Math.round((f_in[c_index]-
(float) fp_min)*temp2);
         int ie=Math.round((f_in[e_index]-
(float) fp_min)*temp2);
         uk[i]=tabelaAfin[ic*N256 + ie];
         flmin=(int) (MAX_AFIN * uk[i]);
         if(f_c < flmin) flmin=f_c;

/**
* 7)     if (fmin > f[e] then
* 8)         //set f[e]=fmin
* 9)         //if (e is already in Q) then remove e//update the
* location of e in Q
* 10)                                     //else push e to Q
* 11)         set f[e]=fmin
* 12)         push e to Q
*/

        if(flmin > fconn[e_index]){
            fconn[e_index]=flmin;
            lk=Q.elementAt(fconn[e_index]);
            lk.add(e);
            Q.setElementAt(lk, fconn[e_index]);
        }

    }

    for(int i=0; i< nSpels; i++)
f_out[i]=(float) fconn[i]/MAX_AFIN;
return(f_out);
}

// ===== metodo =====
// identida da classe Spel p/ aplicar em tabela
float getAfinidadePorValor(float fc, float fd,
                           double m1, double s1, double m2, double s2, double
lambda) {
    double g1=(Math.abs(fc-fd) - m1)/s1;
    g1=g1*g1;
    double g2=((fc+fd)/2 - m2)/s2;
    g2=g2*g2;
    return((float)Math.exp(-(g1+g2)/lambda));
}

// ===== metodo =====
void inicializaFuzzySet(float[] f, Vector<Spel> seeds) {
    int nSpels=nVol*nSli*nLin*nCol;

```

```

        for (int p = 0; p < nSpels; p++)
            f[p]=(float)0.0
        for (int i = 0; i < seeds.size(); i++)
        {
            int c = seeds.elementAt(i).getIndex();
            f[c]=(float)1.0;
        }
    }
    // ===== metodo =====
    void inicializaFuzzySet(int[] f, Vector<Spel> seeds, int
maxAfin) {
        int nSpels=nVol*nSli*nLin*nCol;

        for (int p = 0; p < nSpels; p++)
            f[p]=0;
        for (int i = 0; i < seeds.size(); i++)
        {
            int c = seeds.elementAt(i).getIndex();
            f[c]=maxAfin;
        }
    }
    // ===== metodo =====
    private void inicializaFuzzySet(byte[] f, Vector<Spel> seeds) {
        int nSpels=nVol*nSli*nLin*nCol;
        for (int p = 0; p < nSpels; p++)
            f[p]=0;
        for (int i = 0; i < seeds.size(); i++)
        {
            int c = seeds.elementAt(i).getIndex();
            f[c]=1; //f.set(c, 1);
        }
    }
    // ===== metodo =====
    /**
    * Obter caracteristicas das sementes (m1,s1), (m2,s2)
    * Verificado em imagens do tipo 8G,8C,16G, homogeneo e hetero.
    */

    private double[] obter_m1s1_m2s2_2D(Vector<Spel>
seeds,double sigma_min, boolean IJ_presente) {
        double[] m1s1m2s2=new double[4];
        int[] rows = new int[4];
        int[] cols = new int[4];
        double m1 = 0, s1 = 0, m2 = 0, s2 = 0;
        int r,c,N = 0;
        Spel s;

        for(int k=0; k< seeds.size(); k++){
            s=seeds.elementAt(k);
            c=s.col;
            r=s.lin;
            rows[0] = r - 1;
            cols[0] = c;
            rows[1] = r;
            cols[1] = c + 1;

```

```

        rows[2] = r + 1;
        cols[2] = c;
        rows[3] = r;
        cols[3] = c - 1;
        double f0 =f_in[s.getIndex()];
        int offset=s.vol*(nSli*nLin*nCol) + s.sli *
(nLin*nCol);
        for (int i = 0; i < 4; i++) {
            double temp =f_in[offset + rows[i]*nCol +
cols[i]];

            m1 += Math.abs(f0 - temp);
            s1 += (f0 - temp) * (f0 - temp);
            m2 += (f0 + temp) / 2;
            s2 += (f0 + temp) * (f0 + temp) / 4;
            N++;
        }
        m1 = m1 / N;
        m2 = m2 / N;
        s1 = s1 / N - m1 * m1;          s1 = Math.sqrt(s1);
        s2 = s2 / N - m2 * m2;          s2 = Math.sqrt(s2);
        boolean AVISAR = false;
        if (s1 < sigma_min) {
            s1 = sigma_min;
            AVISAR = true;
            if(IJ_presente) IJ.showMessage("s1 muito
pequeno" + IJ.d2s(s1) + ". Set p/ "
+ IJ.d2s(s1));
        }
        if (s2 < sigma_min) {
            s2 = sigma_min;
            AVISAR = true;
            if(IJ_presente) IJ.showMessage("s2 muito
pequeno" + IJ.d2s(s2) + ". Set p/ "
+ IJ.d2s(s2));
        }
        if (AVISAR)
            if(IJ_presente) IJ.write("Depois de check de
sigma_min: \n (m1,s1)=( " + IJ.d2s(m1)
+ ", " + IJ.d2s(s1) + ") \n
(m2,s2)=( " + IJ.d2s(m2) + ", "
+ IJ.d2s(s2) + ")");
        m1s1m2s2[0] = m1;
        m1s1m2s2[1] = s1;
        m1s1m2s2[2] = m2;
        m1s1m2s2[3] = s2;
        return(m1s1m2s2);
    }
}

import java.util.Vector;
import ij.ImageStack;

/*****
* classe Spel - coordenadas do spel.

```

```

*****/

public class Spel {
    int col, lin, sli, vol;
    private ImageStack stack;

    Spel(int col, int lin, int sli, int vol, ImageStack stack) {
        this.col=col;
        this.lin=lin;
        this.sli=sli;
        this.vol=vol;
        this.stack=stack;
    }

    // obtem o indice na matriz n-d

    int getIndex( ){
        int nSli=stack.getSize(); int nLin=stack.getHeight();
        int nCol=stack.getWidth();
        return
        (vol*(nSli*nLin*nCol)+sli*(nLin*nCol)+lin*nCol+col);
    }
    float getValue(float[] matrizNd){
        return (matrizNd[this.getIndex()]);
    }

    /**obtem colecao de spels adjacentes nD
    * considera adjacente MESMO se vizinho tiver valor fuzzy fl=1
    */

    Vector<Spel> getAdjacentesNd(){
        int nSli=stack.getSize(); int nLin=stack.getHeight();
        int nCol=stack.getWidth();
        int nVizinhos=4; // 2D
        if (nSli > 1) nVizinhos=6; // 3d
        // if (nVol > 1 ) nVizinhos = 8; // 4d ATTENTION
        Vector<Spel> Q=new Vector<Spel>(nVizinhos);
        //Norte No mesmo plano
        int r=lin-1; int c=col;
        if(r>=0 ) Q.add(new Spel(c,r,sli,vol,stack));
        //S
        r=lin+1;
        if(r<nLin ) Q.add(new Spel(c,r,sli,vol,stack));
        //E
        r=lin; c=col+1;
        if(c<nCol ) Q.add(new Spel(c,r,sli,vol,stack));
        //O
        c=col-1;
        if(c>=0 ) Q.add(new Spel(c,r,sli,vol,stack));

        if(nSli>1){// checando slices anteriores e posteriores
            int s=sli-1;
            if (s >=0) Q.add(new Spel(col, lin,s,vol,stack));
            s=sli+1;
            if (s < nSli) Q.add(new Spel(col, lin,s,vol,stack));
        }
    }
}

```

```

        return (Q);
    }

    /**obtem colecao de spels adjacentes
     * Nao considera adjacente se vizinho tiver valor fuzzy fl=1
     */
    Vector<Spel> getAdjacentesNdNaoSementes(float[] fuzzy){
        double epsilon=1.0e-4;
        int nSli=stack.getSize();
        int nLin=stack.getHeight();
        int nCol=stack.getWidth();
        int nVizinhos=4; // 2D
        if (nSli > 1) nVizinhos=6; // 3d
        // if (nVol >1 ) nVizinhos = 8; // 4d ATTENTION
        Vector<Spel> Q=new Vector<Spel>(nVizinhos);
        //Norte No mesmo plano
        int ind_plano=vol*(nSli*nLin*nCol) + sli*(nLin*nCol);
        int r=lin-1; int c=col; int index= ind_plano + r*nCol + c;
        if(r>=0 &&(fuzzy[index] < (1.0-epsilon))) Q.add(new
        Spel(c,r,sli,vol,stack));
        //S
        r=lin+1; index= ind_plano + r*nCol + c;
        if(r<nLin &&(fuzzy[index] < (1.0-epsilon))) Q.add(new
        Spel(c,r,sli,vol,stack));
        //E
        r=lin; c=col+1; index= ind_plano + r*nCol + c;
        if(c<nCol &&(fuzzy[index] < (1.0-epsilon))) Q.add(new
        Spel(c,r,sli,vol,stack));
        //O
        c=col-1; index= ind_plano + r*nCol + c;
        if(c>=0 &&(fuzzy[index] < (1.0-epsilon))) Q.add(new
        Spel(c,r,sli,vol,stack));

        if(nSli>1){// checando slices anteriores e posteriores
            int s=sli-1; ind_plano=vol*(nSli*nLin*nCol)+
            lin*nCol + col ;
            index=ind_plano + s*(nLin*nCol);
            if (s >=0 &&(fuzzy[index] < (1.0-epsilon)))
            Q.add(new Spel(col, lin,s,vol,stack));
            s=sli+1; index=ind_plano + s*(nLin*nCol);
            if (s < nSli &&(fuzzy[index] < (1.0-epsilon)))
            Q.add(new Spel(col, lin,s,vol,stack));
        }
        // if(nVol>1)// checando volumes anteriores e posteriores
        ATTENTION

        return (Q);
    }

    /**obtem colecao de spels adjacentes
     * Nao considera adjacente se vizinho tiver valor fuzzy fl != 0
     */
    Vector<Spel> getAdjacentesNdNaoSementes(int[] fuzzy){
        int nSli=stack.getSize(); int nLin=stack.getHeight(); int
        nCol=stack.getWidth();
        int nVizinhos=4; // 2D

```



```

        if (nSli > 1) nVizinhos=6; // 3d
        // if (nVol >1 ) nVizinhos = 8; // 4d ATTENTION
        Vector<Spel> Q=new Vector<Spel>(nVizinhos);
        //Norte No mesmo plano
        int ind_plano=vol*(nSli*nLin*nCol) + sli*(nLin*nCol);
        int r=lin-1; int c=col; int index= ind_plano + r*nCol + c;
        if(r>=0 &&(fuzzy[index] ==0)) Q.add(new
Spel(c,r,sli,vol,stack));
        //S
        r=lin+1; index= ind_plano + r*nCol + c;
        if(r<nLin &&(fuzzy[index] ==0)) Q.add(new
Spel(c,r,sli,vol,stack));
        //E
        r=lin; c=col+1; index= ind_plano + r*nCol + c;
        if(c<nCol &&(fuzzy[index] ==0)) Q.add(new
Spel(c,r,sli,vol,stack));
        //O
        c=col-1; index= ind_plano + r*nCol + c;
        if(c>=0 &&(fuzzy[index] ==0)) Q.add(new
Spel(c,r,sli,vol,stack));

        if(nSli>1){// checando slices anteriores e posteriores
            int s=sli-1; ind_plano=vol*(nSli*nLin*nCol)+
lin*nCol + col ;
            index=ind_plano + s*(nLin*nCol);
            if (s >=0 &&(fuzzy[index] ==0)) Q.add(new Spel(col,
lin,s,vol,stack));
            s=sli+1; index=ind_plano + s*(nLin*nCol);
            if (s < nSli &&(fuzzy[index] ==0)) Q.add(new
Spel(col, lin,s,vol,stack));
        }

        return (Q);
    }
}
/**obtem colecao de spels adjacentes da imagem "matrizNd"
* considerando a conectividade
* "fuzzy" (binaria) corrente
* Nao considera adjacente se vizinho tiver valor fuzzy fl=1
* ou se afinidade < "teta"
*/
Vector<Spel> getAdjacentesNdNaoSementesTetaAfim(float[]
matrizNd, byte[] fuzzy, double teta,
double m1, double s1, double m2, double s2, double
lambda){

    int nSli=stack.getSize(); int
nLin=stack.getHeight(); int nCol=stack.getWidth();
    double uk;
    int nVizinhos=4; // 2D
    if (nSli > 1) nVizinhos=6; // 3d
    Vector<Spel> Q=new Vector<Spel>(nVizinhos);
    float fc=matrizNd[this.getIndex()];
    float fd;
    //Norte
    int ind_plano=vol*(nSli*nLin*nCol) +
sli*(nLin*nCol);

```

```

        int r=lin-1; int c=col; int index= ind_plano +
r*nCol + c;
        if(r>=0 &&(fuzzy[index] != 1)) { fd=matrizNd[index];
            uk=this.getAfinidadeComWeight(fc,fd, m1, s1,
m2, s2, 0.8);
            if(uk >= teta) Q.add(new
Spel(c,r,sli,vol,stack));
        }
        //S
        r=lin+1; index= ind_plano + r*nCol + c;
        if(r<nLin &&(fuzzy[index] !=1)) {fd=matrizNd[index];
            uk=this.getAfinidadeComWeight(fc,fd, m1, s1,
m2, s2, 0.8);
            if(uk >= teta) Q.add(new
Spel(c,r,sli,vol,stack));
        }
        //E
        r=lin; c=col+1; index= ind_plano + r*nCol + c;
        if(c<nCol &&(fuzzy[index] !=1)) {fd=matrizNd[index];
            uk=this.getAfinidadeComWeight(fc,fd, m1, s1, m2,
s2, 0.8);
            if(uk >= teta) Q.add(new
Spel(c,r,sli,vol,stack));
        }

        //O
        c=col-1; index= ind_plano + r*nCol + c;
        if(c>=0 &&(fuzzy[index] !=1)) {fd=matrizNd[index];
            uk=this.getAfinidadePorValor(fc,fd, m1, s1, m2,
s2, 0.8);
            if(uk >= teta) Q.add(new
Spel(c,r,sli,vol,stack));
        }

        if(nSli>1){
            int s=sli-1; ind_plano=vol*(nSli*nLin*nCol)+
lin*nCol + col ;
            index=ind_plano + s*(nLin*nCol);
            if (s >=0 &&(fuzzy[index]
!=1)){fd=matrizNd[index];
                uk=this.getAfinidadeComWeight(fc,fd, m1,
s1, m2, s2, 0.8);
                if(uk >= teta) Q.add(new Spel(col,
lin,s,vol,stack));
            }
            s=sli+1; index=ind_plano + s*(nLin*nCol);
            if (s < nSli &&(fuzzy[index] !=1))
{fd=matrizNd[index];
                uk=this.getAfinidadePorValor(fc,fd, m1,
s1, m2, s2, 0.8);
                Q.add(new Spel(col, lin,s,vol,stack));
            }
        }
        return (Q);
    }

```

```

/**afinidade entre (this c,d)
 * uk(c,d)=afin(c,d)=exp((-1/lambda)*g )
 *      g=[(|f(c)-f(d)|-m1)/s1]^2 + [((f(c)+f(d))/2-m2)/s2]^2
 * @param d: Spel com a qual se calcula a afinidade
 * @return : afinidade
 */
double getAfinidadePorValor(float fc, float fd,
                           double m1, double s1, double m2, double s2, double
lambda) {
    double g1=(Math.abs(fc-fd) - m1)/s1;
    g1=g1*g1;
    double g2=((fc+fd)/2 - m2)/s2;
    g2=g2*g2;
    return(Math.exp(-(g1+g2)/lambda));
}
/**afinidade entre (this c,d) com peso w2 fornecido
 * uk(c,d)=afin(c,d)=w1.g1(c,d)+w2.g2(c,d)
 *      g1=exp{-[ (|f(c)-f(d)|-m1)/s1]^2}
 *      g2=exp{-[ ((f(c)+f(d))/2-m2)/s2]^2}
 *      w1=1-w2
 * @param d: Spel com a qual se calcula a afinidade
 * @return : afinidade (OBS.: w2 entre [0, 1]
 */
double getAfinidadeComWeight(float fc, float fd,
                             double m1, double s1, double m2, double s2, double
w2) {
    double g1=(Math.abs(fc-fd) - m1)/s1;
    g1=g1*g1;
    g1=Math.exp(-g1/2.0);
    double g2=((fc+fd)/2 - m2)/s2;
    g2=g2*g2;
    g2=Math.exp(-g2/2.0);
    double w1=1-w2;
    return(w1*g1+w2*g2);
}

/**afinidade entre (this c,d) com peso dinamico
 * uk(c,d)=afin(c,d)=w1.g1(c,d)+w2.g2(c,d)
 *      g1=exp{-[ (|f(c)-f(d)|-m1)/s1]^2}
 *      g2=exp{-[ ((f(c)+f(d))/2-m2)/s2]^2}
 *      w1=g1/(g1+g2)
 *      w2=g2/(g1+g2)
 *      if(g2<0.01) {w1=w2=0}
 * @param d: Spel com a qual se calcula a afinidade
 * @return : afinidade (OBS.: nao implementei (m1,s1) direcional
 */
double getAfinidadeDynamicWeight(float fc, float fd,
                                  double m1, double s1, double m2, double s2) {
    double g1=(Math.abs(fc-fd) - m1)/s1;
    g1=g1*g1;
    g1=Math.exp(-g1/2.0);
    double g2=((fc+fd)/2 - m2)/s2;
    g2=g2*g2;
    g2=Math.exp(-g2/2.0);
    double w2=g2/(g1+g2);
    double w1=1-w2;

```

```

        if(g2<0.01) {w1=0.0; w2=0.0;}
        return(w1*g1+w2*g2);
    }
}

import ij.IJ;
import ij.ImagePlus;
import ij.ImageStack;
import ij.plugin.filter.PlugInFilter;
import ij.process.FloatProcessor;
import ij.process.ImageProcessor;
import java.awt.Rectangle;
import java.util.Vector;
import java.util.Calendar;

/*****
* OBJETIVO: segmentar imagem 3D (8G,8C,short,float) usando fuzzy
* connectedness (Algoritmo Genérico)
*****/

public class Segmentar_kFOEMS implements PlugInFilter {
    ImagePlus imp;
    protected ImageStack stack;

    // constantes configuraveis:
    private boolean IJresults_presente=true;
    private boolean TESTE_COM_ROI_30x30= false;

    // setup
    public int setup(String arg, ImagePlus imp) {
        this.imp = imp;
        this.stack=imp.getStack();
        return DOES_8G + DOES_8C + DOES_16 + DOES_32 +
NO_CHANGES + ROI_REQUIRED;
    }

    public void run(ImageProcessor ip) {
        int nVol=1;
        int nSli=stack.getSize();  int nLin=stack.getHeight();
int nCol=stack.getWidth();
        int nSpels=nVol*nSli*nLin*nCol;
        float[] f_conn;
        float[] f_in=new float[nSpels];
        for(int sli=0; sli< nSli; sli++){
            FloatProcessor f_temp=(FloatProcessor)
stack.getProcessor(sli+1).convertToFloat();
            int offset=sli*nLin*nCol;
            for(int index=0; index< nLin*nCol; index++)
                f_in[offset+index]=f_temp.getf(index);
        }

        /** 0) Sementes em S de um mesmo objeto fornecidos pelo usuario
        * Obtendo
        * características das sementes: media e dp, (m1,s1) do |f(c)-f(d)|
        * (m2,s2) do (f(c)+f(d))/2 onde (c,d) são adjacentes. P/ sementes
        * usar

```

[illegible]

```

        calendario.get(Calendar.MILLISECOND)+"ms");
        tempoInicio=calendario.getTimeInMillis() - tempoInicio;
        IJ.write("tempo de operacao (nao exclusiva) desde o inicio do
Segmentador \n em (ms): "+tempoInicio);
        // visualizando
        ImageStack stack_f_conn=new ImageStack(nCol,nLin,nSli);
        float[] pixels; //new float[nLin*nCol];
        for(int i=0; i< nSli; i++) {
            int offset=i*nLin*nCol;
            pixels=new float[nLin*nCol];
            for(int j=0; j< nLin*nCol; j++) {
                pixels[j]=f_conn[offset+j];
            }
            stack_f_conn.setPixels((float[])pixels, i+1);
        }
        ImagePlus fl_imp=new ImagePlus("FuzzyConnectedness:
Segmentar_kFOEMS.java",stack_f_conn);
        fl_imp.setDisplayRange(0.0, 1.0);
        fl_imp.show();
        fl_imp.updateAndDraw();
        IJ.write("Fim do processamento ----->>>>>>>>>>>>>>>>>>>\n");
    }
}

import ij.IJ;
import ij.ImagePlus;
import ij.ImageStack;
import ij.plugin.filter.PlugInFilter;
import ij.process.FloatProcessor;
import ij.process.ImageProcessor;
import java.awt.Rectangle;
import java.util.Vector;
import java.util.Calendar;

/*****
* Segmentar_Dial_LIFO
* Implementação baseado no artigo do Nyul
*****/

public class Segmentar_Dial_LIFO implements PlugInFilter {
    ImagePlus imp;
    protected ImageStack stack;

    private boolean IJresults_presente=true;
    private boolean TESTE_COM_ROI_30x30= false;
    private boolean LIFO=true;
    public int setup(String arg, ImagePlus imp) {
        this.imp = imp;
        this.stack=imp.getStack();
        return DOES_8G + DOES_8C + DOES_16 + DOES_32 +
NO_CHANGES + ROI_REQUIRED;
    }

    public void run(ImageProcessor ip) {
        int nVol=1;

```

```

        int nSli=stack.getSize();  int nLin=stack.getHeight();
int nCol=stack.getWidth();
        int nSpels=nVol*nSli*nLin*nCol;
        float[] f_conn;
        float[] f_in=new float[nSpels];
        for(int sli=0; sli< nSli; sli++){
            FloatProcessor f_temp=(FloatProcessor)
stack.getProcessor(sli+1).convertToFloat();
            int offset=sli*nLin*nCol;
            for(int index=0; index< nLin*nCol; index++)
                f_in[offset+index]=f_temp.getf(index);
        }

/** 0) Sementes em S de um mesmo objeto fornecidos pelo usuario
* Obtendo
* características das sementes: media e dp, (m1,s1) do |f(c)-f(d)|
* e (m2,s2) do (f(c)+f(d))/2 onde (c,d) são adjacentes. P/
* sementes usar
* adjacencia 2D e 4. --check se s1 ou s2 muito pequenos. Compare
* com (fmax - fmin)/100 (40dB)
*/

        int slice=imp.getCurrentSlice()-1;
        Rectangle roi_s=stack.getRoi();
        IJ.showMessage("Atenção", "Assume que o ROI é a semente.
Slice[0.." + (nSli-1)
                    +"]:"+slice);

        // testando
        if(TESTE_COM_ROI_30x30){
            roi_s.setBounds(100, 100, 30, 30);
            IJ.showMessage("Usando ROI pre-definido em (100,100),
30x30");
        }

        int r0 = roi_s.y, h = roi_s.height;
        int c0 = roi_s.x, w = roi_s.width;
        if (h <= 2 | w <= 2) {
            IJ.showMessage("Atencao:", "ROI muito pequeno");
            return;
        }
        Vector<Spel> sementes=new Vector<Spel>(h * w);
        for (int r = r0 ; r < r0 + h ; r++)
            for (int c = c0 ; c < c0 + w ; c++)
                {sementes.add(new Spel( c, r, slice, 0, stack)); }
        // Marcando o tempo de maquina
        Calendar calendario=Calendar.getInstance();
        long tempoInicio=calendario.getTimeInMillis();
        IJ.write("<<<<<<<<<----- Programa (classe)
Segmentar_Dial_LIFO.java (com Tabela Afins) \n" +
                "[memoria disponivel p/ ImageJ=
"+IJ.maxMemory()/1e6 +" MB]");
        if(LIFO) IJ.write(" Opcao de agregacao de vizinho em
caso de EMPATE: LIFO");
        else IJ.write(" Opcao de agregacao de vizinho em
caso de EMPATE: FIFO");

```

```
IJ.write("***Inicio no dia:"+
calendario.get(Calendar.DATE)+"às
"+calendario.get(Calendar.HOUR_OF_DAY)

+h"+calendario.get(Calendar.MINUTE)+"'"+calendario.get(Cale
ndar.SECOND)+".."+
        calendario.get(Calendar.MILLISECOND)+"ms");
// <<<<<<<<<<<<<<<< INICIO do processamento
//chamando o segmentador ....
Segmentador segm=new Segmentador(stack,f_in );
// - EXECUTANDO a segmentacao
double lambda=4.0;
f_conn=segm.Dijkstra_Dial_comTabelaAfin(sementes,lambda,
IJresults_presente, LIFO);
// FIM do processamento

// Marcando o tempo final
calendario=Calendar.getInstance();
IJ.write("***Fim do processamento no dia:"+
calendario.get(Calendar.DATE)+"às
"+calendario.get(Calendar.HOUR_OF_DAY)

+h"+calendario.get(Calendar.MINUTE)+"'"+calendario.get(Cale
ndar.SECOND)+".."+
        calendario.get(Calendar.MILLISECOND)+"ms");
tempoInicio=calendario.getTimeInMillis() - tempoInicio;
IJ.write("tempo de operacao (nao exclusiva)desde o inicio do
Segmentador \n em (ms):"+tempoInicio);
// visualizando
ImageStack stack_f_conn=new ImageStack(nCol,nLin,nSli);
float[] pixels;
for(int i=0; i< nSli; i++) {
    int offset=i*nLin*nCol;
    pixels=new float[nLin*nCol];
    for(int j=0; j< nLin*nCol; j++) {
        pixels[j]=f_conn[offset+j];
    }
    stack_f_conn.setPixels((float[])pixels, i+1);
}
ImagePlus fl_imp=new ImagePlus("FuzzyConnectedness:
Segmentar_Dial_LIFO.java",stack_f_conn);
fl_imp.setDisplayRange(0.0, 1.0);
fl_imp.show();
fl_imp.updateAndDraw();
IJ.write("Fim do processamento ----->>>>>>>>>>>>>>>\n");
}
```



```

import java.util.Vector;
import java.util.Calendar;

/*****
 * Segmentacao usando o algoritmo kTetaFOEMS
 * Udupa. Proc. IEEE, vol 91, n10, 2003, p.1649-1669
 *****/

public class Segmentar_kTetaFOEMS implements PlugInFilter {
    ImagePlus imp;
    protected ImageStack stack;

    private boolean IJresults_presente=true;
    private boolean TESTE_COM_ROI_30x30= false;
    public int setup(String arg, ImagePlus imp) {
        this.imp = imp;
        this.stack=imp.getStack();
        return DOES_8G + DOES_8C + DOES_16 + DOES_32 +
NO_CHANGES + ROI_REQUIRED;
    }

    public void run(ImageProcessor ip) {
        int nVol=1; // ATTENTION
        int nSli=stack.getSize(); int nLin=stack.getHeight();
int nCol=stack.getWidth();
        int nSpels=nVol*nSli*nLin*nCol;
        byte[] f_conn;
        float[] f_in=new float[nSpels];
        for(int sli=0; sli< nSli; sli++){
            FloatProcessor f_temp=(FloatProcessor)
stack.getProcessor(sli+1).convertToFloat();
            int offset=sli*nLin*nCol;
            for(int index=0; index< nLin*nCol; index++)
                f_in[offset+index]=f_temp.getf(index);
        }
        double teta=0.1;
        teta=IJ.getNumber("Valor de TETA=", 0.1);

    /** 0) Sementes em S de um mesmo objeto fornecidos pelo usuario
    * Obtendo
    * características das sementes: media e dp, (m1,s1) do |f(c)-f(d)|
    * e (m2,s2) do (f(c)+f(d))/2 onde (c,d) são adjacentes. P/ sementes
    * usar adjacencia 2D e 4. --check se s1 ou s2 muito pequenos.
    * Compare com (fmax - fmin)/100 (40dB)
    */
        Rectangle roi_s=stack.getRoi();
        int slice=imp.getCurrentSlice()-1;
        IJ.showMessage("Atenção", "Assume que o ROI é a semente.
Slice[0.." + (nSli-1)
        +"]:" + slice + ". Teta=" + teta);

        // testando
        if(TESTE_COM_ROI_30x30){
            roi_s.setBounds(100, 100, 30, 30);
            IJ.showMessage("Usando ROI pre-definido em (100,100),
30x30");
        }
    }
}

```

[illegible]

```

        stack_f_conn.setPixels((float[])pixels, i+1);
    }
    ImagePlus f1_imp=new ImagePlus("FuzzyConnectedness:
Segmentar_kTetaFOEMS.java",stack_f_conn);
    f1_imp.setDisplayRange(0.0, 1.0);
    f1_imp.show();
    f1_imp.updateAndDraw();

    int[] rgb;
    ImageStack rgb_stack=new ImageStack(nCol,nLin,nSli);
    float fp_min, fp_max;
    fp_min=fp_max= f_in[0];
    for(int i=0; i<f_in.length; i++) {
        float temp=f_in[i];
        if(temp > fp_max) fp_max=temp;
        else if(temp < fp_min) fp_min=temp;
    }
    for(int sli=0; sli<nSli; sli++){
        int offset=sli*nLin*nCol;
        rgb=new int[nLin*nCol];
        for(int i=0; i < nLin*nCol; i++){
            int cor=(int) (f_in[offset+i]/(fp_max-
fp_min)*255) & 0xff;
            rgb[i]=cor << 16 | cor << 8 | cor;
            if(f_conn[offset+i] == 1) {
                rgb[i]=rgb[i] & 0xff0000;
            }
        }
        rgb_stack.setPixels(rgb, sli+1);
    }
    ImagePlus rgb_imp=new ImagePlus("sobrepastos",rgb_stack);
    rgb_imp.show();
    IJ.write("Fim do processamento ----->>>>>>>>>>>>>>>>>>>>>>>>>\n");
}

import ij.IJ;
import ij.ImagePlus;
import ij.ImageStack;
import ij.plugin.filter.PlugInFilter;
import ij.process.FloatProcessor;
import ij.process.ImageProcessor;
import java.util.Vector;
import java.util.Calendar;

/*****
* OBJETIVO: segmentar imagem nD (8G,8C,short,float)
* usando o dynamic weight
*****/

public class Segmentar_kFOEMS_DyW implements PlugInFilter {
    ImagePlus imp;
    protected ImageStack stack;

    private boolean IJresults_presente=true;
    private boolean TESTE_COM_ROI_30x30= false;

```

```

    public int setup(String arg, ImagePlus imp) {
        this.imp = imp;
        this.stack=imp.getStack();
        return DOES_8G + DOES_8C + DOES_16 + DOES_32 +
NO_CHANGES + ROI_REQUIRED;
    }

    public void run(ImageProcessor ip) {
        int nVol=1;
        int nSli=stack.getSize(); int nLin=stack.getHeight();
int nCol=stack.getWidth();
        int nSpels=nVol*nSli*nLin*nCol;
        float[] f_conn;
        float[] f_in=new float[nSpels];
        for(int sli=0; sli< nSli; sli++){
            FloatProcessor f_temp=(FloatProcessor)
stack.getProcessor(sli+1).convertToFloat();
            int offset=sli*nLin*nCol;
            for(int index=0; index< nLin*nCol; index++)
                f_in[offset+index]=f_temp.getf(index);
        }

/** 0) Sementes em S de um mesmo objeto fornecidos pelo usuario
* Obtendo
* características das sementes: media e dp, (m1,s1) do |f(c)-f(d)|
* e (m2,s2) do (f(c)+f(d))/2 onde (c,d) são adjacentes. P/ sementes
* usar adjacencia 2D e 4. --check se s1 ou s2 muito pequenos.
* Compare com (fmax - fmin)/100 (40dB)
*/
        Rectangle roi_s=stack.getRoi();
        int slice=imp.getCurrentSlice()-1;
        IJ.showMessage("Atenção", "Assume que o ROI é a semente.
Slice[0.." + (nSli-1)
            +"]:" + slice);

        // testando
        if(TESTE_COM_ROI_30x30){
            roi_s.setBounds(100, 100, 30, 30);
            IJ.showMessage("Usando ROI pre-definido em (100,100),
30x30");
        }

        int r0 = roi_s.y, h = roi_s.height;
        int c0 = roi_s.x, w = roi_s.width;
        if (h <= 2 | w <= 2) {
            IJ.showMessage("Atencao:", "ROI muito pequeno");
            return;
        }
        Vector<Spel> sementes=new Vector<Spel>(h * w);
        for (int r = r0 ; r < r0 + h ; r++)
            for (int c = c0 ; c < c0 + w ; c++)
                {sementes.add(new Spel( c, r, slice, 0, stack)); }
        // Marcando o tempo de maquina
        Calendar calendario=Calendar.getInstance();
        long tempoInicio=calendario.getTimeInMillis();
        IJ.write("<<<<<<<<<<----- Programa (classe)
Segmentar_kFOEMS_DyW.java\n" +

```



```

import ij.plugin.*;
import ij.process.*;

/*****
 * Algoritmo de Avaliação
 * Baseado no artigo do Udupa:
 * "A Methodology for Evaluating Image Segmentation Algorithms"
 * @author: Edward Flórez Pacheco
 * @version: 3.2
 * @created: 02-Abril-2011
 * Escola Politécnica da Universidade de São Paulo - Brazil
 * Laboratorio de Engenharia Biomedica (LEB)
 *****/

public class Plugin_Avaliador implements PlugIn {

    int backgroundLevel = 0;
    int objectLevel = -1;
    int width;
    int height;
    String dirStd = "I:\\USP_Eng.Biomedica\\MESTRADO 2010-
2011\\TESES_MESTRADO\\Teste_Imagens2011\\Imagens_128X128X128\\(a) I
magens_Padrao\\";
    String dirSeg = "";
    String dirDest = "";

    String prefix = "Image";

    boolean batchProcess = true;
    boolean readAll = false;

    int totSlices = 128;

/*****
 * Dialog of the plugin.
 *****/

    private boolean Dialogue() {

        GenericDialog gd = new GenericDialog("Algoritmo de
Avaliação");

        gd.addNumericField("Object gray level:", objectLevel,
3);
        gd.addNumericField("segm. image background gray
level:", backgroundLevel, 3);
        gd.addStringField("Standard prefix name:", prefix);
        gd.addCheckbox("Batch process", batchProcess);
        gd.addCheckbox("Read all", readAll);
        gd.addNumericField("Total slices (3D):", totSlices,
3);

        gd.showDialog();
        if (gd.wasCanceled())
            return false;
    }

```

```

        prefix = gd.getNextString();
        objectLevel = (int)gd.getNextNumber();
        backgroundLevel = (int)gd.getNextNumber();
        batchProcess = gd.getNextBoolean();
        readAll = gd.getNextBoolean();
        totSlices = (int)gd.getNextNumber();

        dirSeg = IJ.getDirectory("Select image segmented
folder...");
        if (dirSeg == null)
            return false;

        return (true);
    }

    public void run(String arg) {
        if (!Dialogue())
            return;

        Date t0 = new Date();

        if (batchProcess == false) {

            ImagePlus imp = WindowManager.getCurrentImage();
            if (imp==null) {
                IJ.noImage();
                return;
            }
            if (!(imp.getProcessor() instanceof ByteProcessor))
            {
                IJ.showMessage("FC segmentador", "8-bit image or
stack required.");
                return;
            }

            IJ.showMessage("FC segmentador", "Not implemented.
TODO.");

        } else if (readAll == false) {

            for (int cntS = 1; cntS <= totSlices; cntS++) {

                /*****
                * Building the std name
                *****/

                int contTotal=cntS;
                String strTotal = "";
                if (contTotal > 99)
                    strTotal = String.valueOf(contTotal);
                else if (contTotal > 9)
                    strTotal = "0" + String.valueOf(contTotal);
                else
                    strTotal = "00" + String.valueOf(contTotal);
            }
        }
    }
}

```

```

        String padraoName = "Image" + strTotal + ".PNG";

        /*****
        * Read the standard file
        *****/

        System.out.println("Padrao #" + contTotal + " nome= "
+ dirStd+padraoName);
        ImagePlus imgPadrao = IJ.openImage(dirStd+padraoName);
        if (imgPadrao == null)
            continue;
        System.out.println("Found!");

        /*****
        * try to read the segmented image (same name)
        *****/

        ImagePlus imgSeg = null;
        String segName = "";
        try {
            segName = padraoName;

            System.out.println("Segmented #" + contTotal + " nome= " +
dirSeg + segName);
            imgSeg = IJ.openImage(dirSeg + segName);
            if (imgSeg==null)
                continue;
        } catch (Exception e) {

            System.out.println("Erro! File not found: " +
dirSeg+segName);

            continue;
        }

        /*****
        * Compare standard with the segmented
        *****/

        evaluate(imgPadrao, imgSeg, segName, dirDest);

    }

    } else {
        String segName = "";
        String[] list = new File(dirStd).list();
        if (list==null)
            return;
        for (int i=0; i<list.length; i++) {
            if (list[i] == "Thumbs.db")
                continue;

            ImagePlus imgPadrao =
IJ.openImage(dirStd+list[i]);
            if (imgPadrao==null)
                continue;

```



```

        System.out.println("Padrao #" + i + "
nome= " + dirStd+list[i]);

        ImagePlus imgSeg = null;
        try {
            segName = prefix + list[i];
            System.out.println("Segmented #" + i
+ " nome= " + dirSeg + segName);
            imgSeg = IJ.openImage(dirSeg +
segName);

            } catch (Exception e) {
                System.out.println("Erro! File not
found: " + dirSeg+segName);
                continue;
            }

            evaluate(imgPadrao, imgSeg, segName,
dirDest);
        }
    }

    /* time to process */
    Date t1 = new Date();
    System.out.println("time:" + (t1.getTime() -
t0.getTime()) + " ms");
}

/*****
* Função Principal (Compara as duas imagens [Im. Padrão VS. Im.
* Segmentada] e obtém os valores de comparação: TP e FP.
*****/

    void evaluate(ImagePlus imgPadrao, ImagePlus img2, String
label, String dirOut) {
        width = imgPadrao.getWidth();
        height = imgPadrao.getHeight();

        if (img2.getWidth() != width || img2.getHeight() !=
height) {
            IJ.showMessage("Avaliação", "As imagens devem ter a
mesma dimensão! ");
            return;
        }

        byte[] bytesPadrao =
(byte[])imgPadrao.getProcessor().getPixels();
        byte[] bytesImg =
(byte[])img2.getProcessor().getPixels();

        /*
        * Fazendo os cálculos dos valores de TP e FP
        */

        int cnt_total_obj = 0;
        int cnt_total_bkg = 0;
        int cnt_true_positive = 0;

```

```

    int cnt_false_positive = 0;
    int cnt_true_negative = 0;
    int cnt_false_negative = 0;

    objectLevel = discoverObjLevel (imgPadrao);
    System.out.println ("Discovered obj level = " +
objectLevel);

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {

            if (bytesPadrao[toIndex2D(i, j)] ==
objectLevel) {

                cnt_total_obj++;

                if (bytesImg[toIndex2D(i, j)] ==
backgroundLevel) {

                    cnt_false_negative++;

                } else {
                    cnt_true_positive++;

                }
            } else {
                cnt_total_bkg++;

                if (bytesImg[toIndex2D(i, j)] ==
backgroundLevel) {

                    cnt_true_negative++;

                } else {
                    cnt_false_positive++;

                }
            }
        }
    }

    if (cnt_total_obj == 0) {
        cnt_total_obj = 1;
        cnt_true_positive = 0;
    }
    if (cnt_total_bkg == 0) {
        cnt_total_bkg = 1;
        cnt_false_positive = 0;
    }

    /*****
    * Apresentando os resultados
    *****/

    System.out.println (
        /*showMessage("Results - Evaluation",*/

```

```

        label +
        " True-positive: " + cnt_true_positive +
        ", False-positive: " + cnt_false_positive
+
        ", Pixels Object: " + cnt_total_obj +
        ", Pixels Bkg: " + cnt_total_bkg +
        ", True-positive %: " +
((float)((float)cnt_true_positive/(float)cnt_total_obj)*100) +
        ", False-positive %: " +
((float)((float)cnt_false_positive/cnt_total_bkg)*100)
    );

    IJ.log (/*label + ",\t\t" +*/
((float)((float)cnt_true_positive/(float)cnt_total_obj)*100) +
        "\t\t" +
((float)((float)cnt_false_positive/cnt_total_bkg)*100)

    );
}

private int discoverObjLevel(ImagePlus imgPadrao) {
    int obj = 0;

    byte[] bytesPadrao =
(byte[])imgPadrao.getProcessor().getPixels();

    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {

            if (bytesPadrao[toIndex2D(i, j)] < 0) {
                if (bytesPadrao[toIndex2D(i, j)] <
obj)
                    obj = bytesPadrao[toIndex2D(i,
j)];
            } else {
                if (obj >= 0 &&
bytesPadrao[toIndex2D(i, j)] > obj)
                    obj = bytesPadrao[toIndex2D(i,
j)];
            }
        }
    }

    return obj;
}

private int toIndex2D(int x, int y) {
    if (x < 0)
        x = 0;
    else if (x >= width)
        x = width-1;

    if (y < 0)
        y = 0;
    else if (y >= height)

```

```

        y = height-1;

        return ((y*width) + x);
    }
}

```

H.8 – Quantificador de Imagens (3D)

```

/*****
* Algoritmo pronto na pagina de Plugins do Image J
* http://rsbweb.nih.gov/ij/plugins/voxel-counter.html
*
* Este plugin faz a contagem dos voxels de um conjunto de cortes
* ou stack que formam um volumen.
* Como resultado mostra a quantidade de voxels por cada corte,
* a representação em porcentagem do volumen quantificado respecto
* ao volume total (proporção dos voxels), as dimensões do voxel e
* o volume do objeto quantificado (em mm3).
*****/

public class Voxel_Counter implements PlugIn {

    public void run(String arg) {
        ImagePlus imp = WindowManager.getCurrentImage();
        if (imp==null) {
            IJ.noImage();
            return;
        }
        if (!(imp.getProcessor() instanceof ByteProcessor)) {
            IJ.showMessage("Quantificador", "8-bit image or stack
required.");
            return;
        }
        if
        (imp.getProcessor().getMinThreshold()==ImageProcessor.NO_THRESHOLD
        && !binaryImage(imp)) {
            IJ.showMessage("Quantificador de Volumes", "Thresold
ou stack binario é necessário.");
            return;
        }
        Calibration cal = imp.getCalibration();
        double pw = cal.pixelWidth;
        double ph = cal.pixelHeight;
        double pd = cal.pixelDepth;
        cal.pixelWidth = 1.0;
        cal.pixelHeight = 1.0;

        int nslices = imp.getStackSize();
        Roi roi = imp.getRoi();
        long roiCount = 0;
        ResultsTable rt;
        long volumeCount =
        (long)imp.getWidth()*imp.getHeight()*nslices;
        if (roi==null)
            roiCount = volumeCount;
    }
}

```

```

else if (roi.getType()==Roi.RECTANGLE) {
    Rectangle r = roi.getBounds();
    roiCount = (long)r.width*r.height*nslices;
} else {
    IJ.run("Apagando Resultados");
    IJ.run("Ajustando as medições...", "area decimal=2");
    for (int i=1; i<=nslices; i++) {
        imp.setSlice(i);
        IJ.run("Medição");
    }
    rt = Analyzer.getResultsTable();
    roiCount = 0;
    for (int i=0; i<rt.getCounter(); i++)
        roiCount += rt.getValue("Area", i);
}

IJ.run("Apagando resultados");
IJ.run("Ajustando as medições...", "area limite
decimal=2");
for (int i=1; i<=nslices; i++) {
    imp.setSlice(i);
    IJ.run("Medição");
}
cal.pixelWidth = pw;
cal.pixelHeight = ph;
rt = Analyzer.getResultsTable();
double sum = 0;
for (int i=0; i<rt.getCounter(); i++)
    sum += rt.getValue("Area", i);
IJ.write("");
IJ.write("Thresholded voxels: "+(long)sum);
IJ.write("Promedio voxels por cada corte:
"+IJ.d2s(sum/nslices,2));
IJ.write("Total ROI Voxels: "+roiCount);
IJ.write("Fracção do Volume:
"+IJ.d2s((sum*100)/roiCount,2)+"%");
IJ.write("Voxels no stack: "+volumeCount);
if (cal.scaled()) {
    int digits = Analyzer.getPrecision();
    String units = cal.getUnits();
    double scale = pw*ph*pd;
    IJ.write("");
    IJ.write("Tamanho do Voxel: "+IJ.d2s(cal.pixelWidth,
digits) + "x" + IJ.d2s(cal.pixelHeight,
digits)+"x"+IJ.d2s(cal.pixelDepth, digits)+" "+units);
    IJ.write("Volume: "+IJ.d2s(sum*scale,digits)+"
"+units+"^3");
    IJ.write("Promedio volume por cada corte:
"+IJ.d2s(sum*scale/nslices,digits)+" "+cal.getUnits()+"^3");
    IJ.write("Total volume ROI:
"+IJ.d2s(roiCount*scale,digits)+" "+units+"^3");
    IJ.write("Volume do stack:
"+IJ.d2s(volumeCount*scale,digits)+" "+units+"^3");
}
}

```

```
    boolean binaryImage(ImagePlus imp) {
        ImageStatistics stats = imp.getStatistics();
        boolean isBinary =
stats.histogram[0]+stats.histogram[255]==stats.pixelCount;
        if (isBinary) {
            boolean invertedLut = imp.isInvertedLut();
            ImageProcessor ip = imp.getProcessor();
            if (invertedLut)
                ip.setThreshold(0, 0, ImageProcessor.RED_LUT);
            else
                ip.setThreshold(255, 255, ImageProcessor.RED_LUT);
        }
        return isBinary;
    }
}
```

ANEXO I

PUBLICAÇÕES RESULTANTES DESTE TRABALHO

- FLÓREZ PACHECO, E.; FURUIE S.S. **Cuantificación de la Dinámica de Estructuras en imágenes de Medicina Nuclear en la modalidad PET.** Concurso de Proyectos – IEEE INTERCON UNI 2011 (August 8-13, 2011), Lima – Perú. Proceedings of the XVIII International Congress of Electrical, Electronic and Systems Engineering, 15 páginas, 2011.
- FLÓREZ PACHECO, E.; FURUIE S.S. **Quantification of Dynamic Structures in Nuclear Medicine Images in the PET modality.** Workshop of Theses and Dissertations – SIBGRAPI 2011 (August 28-31, 2011), Maceió, Alagoas – Brasil. Proceedings of the XXIV Brazilian Symposium on Computer Graphics and Image Processing, 6 páginas, 2011.
- FLÓREZ PACHECO, E.; FURUIE S.S. **Segmentation of Nuclear Medicine three-dimensional images using Anscombe transformation.** CinC 2011 – Computing in Cardiology (September 18-21, 2011), Zhejiang University, Hangzhou – China. Proceedings of the 38th Annual Scientific Conference of Computing in Cardiology, 4 páginas, 2011.
- Artigo a ser submetido na Revista Brasileira de Engenharia Biomédica. Previsão: Março 2012.