Bernardo Lopes de Sá Azevedo

Licenciatura em Ciências de Engenharia Biomédica

Reconstrução/Processamento de Imagem Médica com GPU em Tomossíntese

Dissertação para obtenção do Grau de Mestre em Engenharia Biomédica

Orientador: Professor Doutor Pedro Almeida, FCUL
Co-orientador: Professor Doutor Nuno Matela, FCUL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro de 2011

Copyright

Agradecimentos

Os meus agradecimentos são dirigidos às pessoas que mais me marcaram nesta jornada académica, mas não são, nem de perto, as únicas pessoas a merecer toda a minha gratidão. Quero então deixar os meus agradecimentos:

Ao Professor Pedro Almeida, pela oportunidade que me deu para poder trabalhar no Instituto de Biofísica e Engenharia Biomédica e pela sua constante boa disposição.

De todas as pessoas que conheci no IBEB, tenho que endereçar um agradecimento especial ao Nuno Matela e ao Nuno Oliveira por toda a ajuda que me deram, sem eles nunca teria tido um nível de conhecimento suficiente no que a algoritmos de reconstrução de imagem diz respeito. Ao Pedro Barata tenho também que agradecer a sua enorme paixão pela programação em paralelo e pela ajuda que me deu nesse campo.

A todas as pessoas que me acompanharam nestes meses no IBEB, quero agradecer a sua camaradagem e também a paciência para me aturar, que eu sei que nem sempre é fácil. Obrigado Guida, Luís, Sérgio, Cascalho, Nuno Matela, Nuno Oliveira, Ricardo, Sofia, Liliana e Beatriz.

Aos Professores Pedro Miranda, Eduardo Ducla-Soares e Alexandre Andrade pela hospitalidade com que me trataram e aos meus colegas da FCT.

Ao Professor Mário Secca, pela sua entrega e dedicação ao Mestrado Integrado em Engenharia Biomédica da FCT-UNL.

Tenho também que deixar um enorme agradecimento aos utilizadores dos fóruns da NVIDIA®, Stackoverflow e da biblioteca Thrust. Sem eles nunca teria conseguido ultrapassar muitas das minhas dúvidas e num campo tão emergente como a programação em CUDA, nem sempre é fácil arranjar a literatura necessária.

Por estes fenomenais cinco anos de curso e por todos os momentos que passámos: João Martins, Pedro Martins, Leonardo Martins, Nuno Fernandes, Mafalda Fernandes, Sérgio Mendes, Sara Gil, Pedro Cascalho, Ana Valente, Fernando Mota, Ana Frazão, Rita Rosa, Rita Carvalho, Carlos Marques, Hugo Pereira, Filipe Catarino, Milene Bação, Susana Gaspar, Susana Martinho, João Sousa, Francisco Venes, Rui Lucena, Bárbara Oliveira, Filipe Rodrigues, Carlos Mota, Luís Ribeiro, Cátia Rocha e a todos os outros que me esqueci.

Aos meus amigos que me acompanham desde há largos anos e sem os quais eu não seria ninguém: Jorge Lucas, Luis Carvalho, Manuel Porto, João Aldeia, João Perfeito, Rui Leitão, Tiago Gageiro, Pedro Vaz, Pedro Teixeira, Diogo Gonçalves, José Lourenço, Ruben Mendes, José Parra, Ricardo Troncão, Pedro Augusto, Rui Miranda, Ricardo Farinha, Ana Serejo, Inês Batista, Mário Almeida, Márcia Mata, Mariana Magalhães, Luis Rodrigues, Diogo Carvalho.

Á minha família, por todo o apoio e por todo o amor que nutrem por mim. Eu sei que sou um privilegiado por ter nascido no seio desta família.

Por último, quero agradecer à Vanessa por todo o amor e apoio ao longo dos últimos 6 anos. Senão fosses tu a aturar as minhas reclamações em tempos mais conturbados da tese, e a incentivar-me continuamente eu nunca estaria onde estou hoje.

Resumo

A Tomossíntese Digital Mamária (DBT) é uma recente técnica de imagem médica tridimensional baseada na mamografia digital que permite uma melhor observação dos tecidos sobrepostos, principalmente em mamas densas. Esta técnica consiste na obtenção de múltiplas imagens (cortes) do volume a reconstruir, permitindo dessa forma um diagnóstico mais eficaz, uma vez que os vários tecidos não se encontram sobrepostos numa imagem 2D.

Os algoritmos de reconstrução de imagem usados em DBT são bastante similares aos usados em Tomografia Computorizada (TC). Existem duas classes de algoritmos de reconstrução de imagem: analíticos e iterativos. No âmbito deste trabalho foram implementados dois algoritmos iterativos de reconstrução: *Maximum Likelihood – Expectation Maximization* (ML-EM) e *Ordered Subsets – Expectation Maximization* (OS-EM). Os algoritmos iterativos permitem melhores resultados, no entanto são computacionalmente muito pesados, pelo que, os algoritmos analíticos têm sido preferencialmente usados em prática clínica. Com os avanços tecnológicos na área dos computadores, já é possível diminuir consideravelmente o tempo que leva para reconstruir uma imagem com um algoritmo iterativo.

Os algoritmos foram implementados com recurso à programação em placas gráficas − *General-Purpose computing on Graphics Processing Units* (GPGPU). A utilização desta técnica permite usar uma placa gráfica (GPU – *Graphics Processing Unit*) para processar tarefas habitualmente designadas para o processador de um computador (CPU – *Central Processing Unit)* ao invés da habitual tarefa do processamento gráfico a que são associadas as GPUs.

Para este projecto foi usado uma GPU NVIDIA®, recorrendo-se à arquitectura Compute Unified Device Architecture (CUDA™) para codificar os algoritmos de reconstrução.

Os resultados mostraram que a implementação dos algoritmos em GPU permitiu uma diminuição do tempo de reconstrução em, aproximadamente, 6,2 vezes relativamente ao tempo obtido em CPU. No respeitante à qualidade de imagem, a GPU conseguiu atingir um nível de detalhe similar às imagens da CPU, apesar de diferenças pouco significativas.

Palavras-chave: Tomossíntese Mamária Digital, Algoritmos Iterativos de Reconstrução, *Maximum Likelihood – Expectation Maximization* (ML-EM), *Ordered Subsets – Expectation Maximization* (OS-EM), *General-purpose computing on Graphics Processing Units (GPGPU), Compute Unified Device Architecture* (CUDA™)

Abstract

Digital Breast Tomosynthesis (DBT) is a tridimensional imaging technique based on digital mammography which allows a better observation of overlapping tissues, mainly in dense breasts. This technique consists in obtaining multiple images (slices) of the volume to be reconstructed, allowing a better diagnosis, since the tissues are not overlapped in a 2D image.

The image reconstruction algorithms used in DBT are similar to the ones used in Computed Tomography (CT). There are two classes of reconstruction algorithms: analytical and iterative. In the scope of this work, two iterative reconstruction algorithms were implemented: Maximum Likelihood – Expectation Maximization (ML-EM) and Ordered Subsets – Expectation Maximization (OS-EM). The iterative algorithms allow better results, however they are computationally intensive so the analytical algorithms have been rather used in clinical practice. With the technologic improvements, particularly in the computers area, it is now possible to substantially decrease the iterative algorithms' reconstruction time.

The algorithms were implemented using graphic cards' programming − General-Purpose computing on Graphics Processing Units (GPGPU). By using this technique it is possible to use a graphic card to perform tasks that are normally held by the processor of a computer – *Central Processing Unit* (CPU), instead of the graphic computation that the GPU is traditionally associated with.

For this project, an NVIDIA® GPU was used, and through the Compute Unified Device Architecture (CUDA™) it was possible to code the reconstruction algorithms.

The results showed that the algorithms implementation in GPU allowed a decrease in the reconstruction times in, approximately 6,2 times compared to the time obtained in CPU. Concerning image quality, the GPU achieved a level of detail similar to the CPU, in spite of minor differences.

Keywords: Digital Breast Tomosynthesis, Iterative Reconstruction Algorithms, Maximum Likelihood – Expectation Maximization (ML-EM), Ordered Subsets – Expectation Maximization (OS-EM), General-purpose computing on Graphics Processing Units (GPGPU), Compute Unified Device Architecture (CUDA™)

Índice

Índice de Figuras

Índice de Tabelas

Siglas e Acrónimos

API	Interface de Programação de Aplicativos (Do inglês *Application Programming Interface*)
ART	Do inglês *Algebraic Reconstruction Technique*
BLAS	Subprogramas básicos de álgebra linear (Do inglês *Basic Linear Algebra Subprograms*)
CAD	Detecção/Diagnóstico assistido por computador (Do *inglês Computer Assisted Detection/Diagnosis*)
CCD	Dispositivos de carga acoplada (Do inglês C*harged Coupled Device*)
CPU	Unidade central de processamento (Do inglês *Central Processing Unit)*
CT	Tomografia computorizada (Do inglês *Computed Tomography*)
CUDA	Do inglês *Compute Unified Device Architecture*
CV	Coeficiente de Variação
DBT	Tomossíntese digital mamária (Do inglês *Digital Breast Tomosynthesis*)
FLOP	Operações de vírgula flutuante por segundo (Do inglês *Floating-Point Operations Per Second*)
FOV	Campo de visão (Do inglês *Field of View*).
FWHM	Largura a meia altura (Do inglês *Full Width at Half Maximum*)
GPU	Unidade de processamento gráfico (Do inglês *Graphics Processing Unit)*
GPGPU	Do inglês *General-purpose computing on Graphics Processing Units*
IDL	Do inglês *Interactive Data Language*
LOR	Linha de resposta (Do inglês *Line of Response*)
ML-EM	Do inglês *Maximum Likelihood – Expectation Maximization*
MRI	Ressonância Magnética (Do inglês *Magnetic Resonance Imaging*)
OS-EM	Do inglês *Ordered Subsets – Expectation Maximization*
PET	Tomografia por Emissão de Positrões (Do inglês *Positron Emission Tomography*)
s-DBT	Tomossíntese digital mamária estacionária (Do inglês *Stationary Digital Breast Tomosynthesis*).
ROI	Região de interesse (Do inglês *Region of Interest*)
SIMD	Instrução única, múltiplos dados (Do inglês *Single Instruction, Multiple Data*)
SM	Multiprocessadores de streaming (Do inglês *Streaming Multiprocessors*)

SNR Razão Sinal-Ruído (Do inglês *Signal-to-Noise Ratio*)

SPECT Tomografia computorizada por emissão de fotão único (Do inglês *Single Positron Emission Computed Tomography*)

STL Do inglês *Standard Template Library*

TFT Transístores de filme fino (Do inglês *Thin-film Transistors*)

US Ultra-sons

Capítulo 1

Introdução

O cancro da mama é a neoplasia maligna mais comum entre as mulheres (afecta uma em cada oito mulheres) e é o segundo cancro com maior taxa de mortalidade sendo apenas suplantado pelo cancro do pulmão [1, 2].

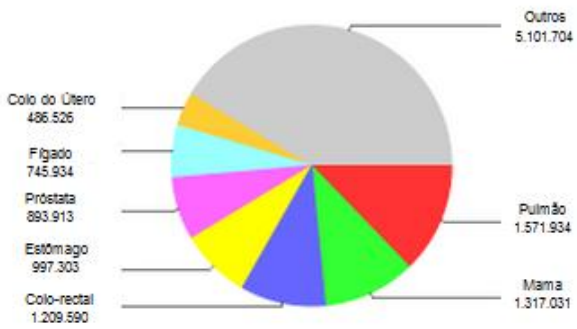

Figura 1.1 Número de casos de cancro diagnosticados, em ambos os sexos, em 2008. Adaptado de [2]

Nas Figuras 1.2 e 1.3 é possível observar as taxas de incidência e de mortalidade do cancro da mama a nível global.

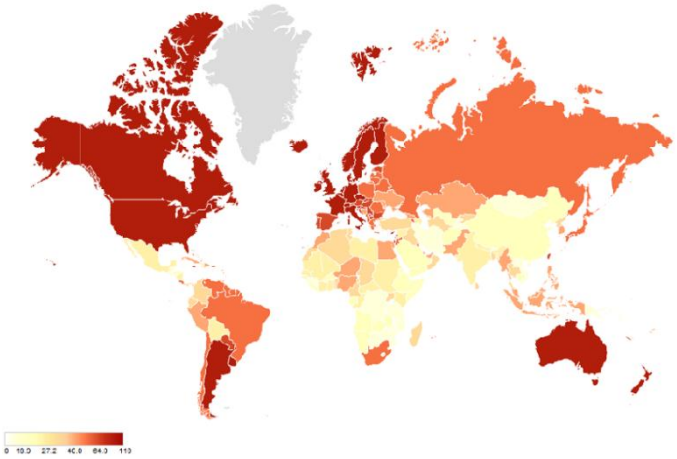

Figura 1.2 Estimativa da taxa de incidência do cancro da mama em 2008 [2].

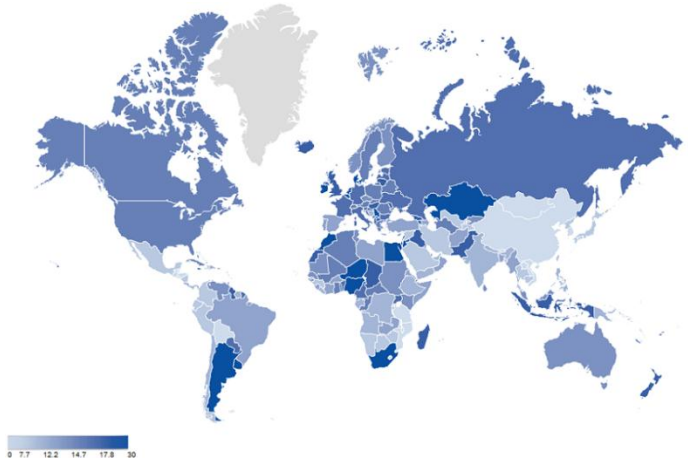

Figura 1.3 Estimativa da taxa de mortalidade do cancro da mama em 2008 [2].

No ano de 2008 morreram, apenas em Portugal, cerca de 1500 pessoas com cancro da mama e as previsões futuras de mortes devidas ao cancro da mama não são animadoras. Prevê-se que no ano de 2015 morram à volta de 1700 pessoas e cinco anos depois, em 2020, esse número chegue perto das 1800 [2].

Devido a estas estatísticas, foram sendo introduzidos programas de rastreio do cancro da mama.

Nas últimas décadas e até aos dias de hoje, a mamografia de raios-X tem sido a principal técnica de imagem médica usada no diagnóstico do cancro da mama, muito devido à sua alta sensibilidade[1] e ao seu custo reduzido. Desde o seu aparecimento, a taxa de incidência do cancro da mama, aumentou significativamente, o que demonstra que passou a existir um rastreio mais eficiente (mas não um aumento factual do número de casos [3]). L. Tabar *et al.* [4] conduziram estudos na Suécia comparando a taxa de mortalidade devido ao cancro da mama, nos 20 anos anteriores e posteriores à introdução de programas de rastreio através de mamografia tendo confirmado uma redução da taxa de mortalidade, como resultado do diagnóstico cada vez mais precoce.

[1] $\text{Sensibilidade} = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$

Devido a certas limitações da mamografia convencional, como o facto de usar radiações ionizantes, sobreposição de tecidos na imagem, desconforto aquando da realização do exame devido à compressão da mama, ou o número elevado de falsos positivos, novas técnicas começaram a ser usadas no diagnóstico do cancro da mama como os ultra-sons (US), ressonância magnética (MRI), tomografia computorizada por emissão de fotão único (SPECT) ou tomografia por emissão de positrões (PET) [5, 6]. É de salientar também o facto de as técnicas de SPECT e PET produzirem imagens funcionais, que permitem observar alterações fisiológicas numa região específica do organismo, ao contrário das imagens anatómicas produzidas pelos raios-X, MRI ou US, nas quais se observam alterações estruturais ou da vascularização (no caso da MRI).

Apenas aprovada pela Food and Drug Administration no corrente ano [7, 8], a DBT é uma técnica baseada na mamografia digital por raios-X. O equipamento utilizado é semelhante ao da mamografia digital, diferindo desta pelo facto de serem feitas várias aquisições da mama comprimida e estacionária, em vários ângulos de exposição. Após um processo de reconstrução de imagem é possível obter uma imagem tridimensional (sendo que na realidade, são obtidos múltiplos cortes do volume e não o volume 3D por inteiro).

Existem duas grandes classes de algoritmos usados em reconstrução de imagem, sendo que os mais usados em prática clínica pertencem à classe dos algoritmos analíticos. A outra classe de algoritmos de reconstrução de imagem, algoritmos iterativos, são computacionalmente muito pesados, o que compromete os seus índices de popularidade. No entanto, com os avanços tecnológicos inerentes à indústria informática, tem sido possível, nos últimos anos, diminuir o tempo de reconstrução destes algoritmos e explorar as suas vantagens relativamente aos algoritmos analíticos (como irá ser discutido numa secção posterior).

Um requisito obrigatório para a implementação de algoritmos iterativos de reconstrução de imagem é a implementação prévia de uma matriz de sistema. A matriz de sistema é um modelo geométrico, independente do objecto a reconstruir (mas dependente da posição angular da ampola de raios-X) que tem como objectivo modular os processos físicos de transmissão e detecção.

O objectivo desta tese é tornar a execução de dois algoritmos iterativos de reconstrução, ML-EM e OS-EM, significativamente mais rápida, tirando partido da emergente programação em paralelo (GPGPU), com recurso a placas gráficas (GPU).

As GPUs eram, até recentemente, usadas apenas para computações gráficas, como jogos de vídeo ou *software* para reprodução de vídeo. No entanto, com o avançar da tecnologia, as GPUs tornaram-se aparelhos altamente programáveis, tendo-lhes sido reconhecida a capacidade para desempenharem funções que eram, até então, normalmente desempenhadas pelo processador de um computador (CPU) [9].

Os resultados finais, quer a imagem obtida, quer o tempo de execução dos algoritmos, foram comparados com um código CPU (código sequencial) desenvolvido na linguagem *Interactive Data Language*® (IDL) para DBT.

Esta tese está dividida em 5 capítulos, sendo o primeiro, esta introdução. O segundo capítulo tratará de oferecer um contexto teórico essencial para a compreensão do trabalho desenvolvido. Os aspectos teóricos fundamentais que irão ser abordados no capítulo 2 incluem: uma breve introdução histórica sobre a descoberta dos raios-X e descrição da técnica de imagem de diagnóstico por raios-X; a técnica de mamografia convencional e digital; a técnica de DBT; reconstrução de imagem e algoritmos de reconstrução e finalmente considerações sobre a GPU e CUDA™.

Serão descritas as vantagens e as desvantagens da DBT relativamente à mamografia convencional e serão também descritos, com rigor, os princípios físicos associados a esta técnica de imagiologia. Na secção dedicada aos algoritmos de reconstrução irão ser abordadas as várias classes de algoritmos, com um maior enfâse na teoria referente aos algoritmos iterativos. Relativamente à GPU, será referido então como esta se tornou um substituto sério na computação de tarefas que, até muito recentemente, eram realizadas pela CPU e por fim serão descritos alguns princípios de funcionamento da arquitectura CUDA™.

No capítulo 3, referente aos Materiais e Métodos, será referido o processo de desenvolvimento da matriz de sistema; uma descrição introdutória da biblioteca Thrust, que permite aproximar a programação em paralelo da programação sequencial; uma explicação do funcionamento básico de um programa em CUDA™ e serão feitas referências a problemas de memória que surgiram aquando da realização deste trabalho, devido à memoria limitada de uma GPU. Os resultados serão analisados no capítulo 4 e serão comparados com um código previamente desenvolvido em CPU.

No derradeiro capítulo serão apresentadas as principais conclusões obtidas através da realização do trabalho e perspectivas de trabalho futuro.

Capítulo 2

Considerações Teóricas

2.1 Imagem Médica e raios-X

Wilhelm Conrad Röntgen, físico da Universidade de Würzburg, foi o responsável pela descoberta dos raios-X, tendo sido ele que lhes atribuiu o nome como são actualmente conhecidos. A primeira imagem de raios-X foi tirada à mão da mulher de Röntgen, decorria o ano 1895, usando apenas uma película fotográfica. Röntgen estava a conduzir experiências com um tubo de Crooke (no qual uma corrente eléctrica pode ser passada de um eléctrodo para outro através de vácuo) quando descobriu que ocorria fluorescência de uma película a alguma distância do tubo. Com o continuar das experiências com os raios-X, observou algumas das suas propriedades, como o poder penetrante em materiais pouco densos, a sua absorção em materiais como o alumínio e a absorção diferenciada em espessuras iguais de vidro contendo diferentes quantidades de chumbo [10].

Figura 2.1 Primeira imagem de raios-X, mostrando a mão (com um anel) da mulher de Röntgen [11].

Após esta descoberta, a sociedade de então tornou-se bastante interessada, havendo reacções diversas relativamente aos raios-X: peças de teatro, publicidade, anedotas e até receio de que os raios-X se tornassem numa invasão de privacidade [12]. Simultaneamente, os raios-X eram estudados em vários países (principalmente a Alemanha, Inglaterra, França e Estados Unidos da América) como instrumentos para diagnóstico médico [13].

A técnica de diagnóstico por raios-X foi amadurecendo e aquando do início da Primeira Guerra Mundial, a maioria dos hospitais norte-americanos já possuía um aparelho para o efeito.

Após a sua descoberta, os raios-X, tiveram avanços significativos, quer ao nível dos das ampolas de raios-X quer também dos detectores. Foram introduzidos detectores com cintiladores e fotomultiplicadores, tendo aumentado de forma significativa a sensibilidade das imagens de raios-X. As ampolas de raios-X também sofreram grandes avanços, uma vez que era essencial controlar o *output* das ampolas, ou seja controlar a energia e intensidade dos raios-X produzidos. Um dos maiores avanços relativamente às ampolas de raios-X foi a descoberta, por Coolidge e Lilienfeld, em 1913, de que a produção de raios-X podia ser controlada aquecendo o cátodo, o que permitiu uma maior estabilidade dos mesmos [12, 13].

2.1.1 Produção de raios-X

Os raios-X são produzidos numa ampola constituída por um cátodo e um ânodo e uma fonte de alta tensão. O cátodo é constituído por um filamento de tungsténio que quando aquecido liberta electrões que são depois acelerados através de uma diferença de potencial numa câmara de vácuo em direcção a um alvo metálico (ânodo – normalmente também composto por tungsténio). O foco por sua vez concentra os electrões produzidos, num ponto focal do ânodo. Os electrões ao embaterem no ânodo produzem raios-X, sendo que menos de 1%, são convertidos para raios-X úteis, enquanto os restantes se perdem em colisões electrónicas produzindo calor que tem que ser removido da ampola. Esse calor pode ser

dissipado com recurso a um circuito de refrigeração ou então recorrendo a um ânodo rotativo [10].

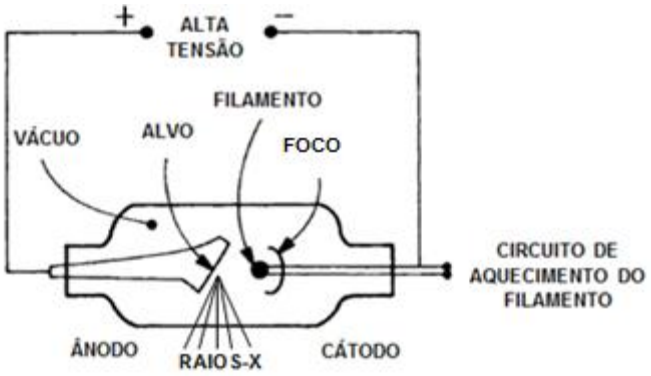

Figura 2.2 Diagrama esquemático de uma ampola de raios-X moderna. Adaptado de [10].

Os raios-X produzidos na ampola, necessitam de ser direccionados para um detector. Para se evitar ruído na imagem ou uma degradação do contraste, é necessário fazer com que os raios-X que não sejam produzidos no ponto focal do ânodo ou que radiação que se encontra na vizinhança do paciente tenham um contributo nulo na imagem final. Os colimadores são assim usados para limitar a forma espacial do feixe de raios-X [14].

A interacção do feixe de raios-X com os tecidos, produz também radiação dispersa que perturba a imagem. Uma forma de eliminar esta radiação é usando uma grelha (composta por tungsténio ou chumbo) imediatamente antes do detector, cuja geometria está alinhada com o ponto focal de origem dos raios-X. A radiação com uma distribuição angular mais dispersa, será absorvida na grelha, não contribuindo para a formação da imagem. No entanto e devido à natureza difusa da radiação dispersa, esta provoca uma intensidade de fundo homogénea o que origina uma diminuição do contraste e da relação sinal-ruído (SNR). Devido ao facto de a grelha diminuir a fluência registada no detector, é necessário aumentar a dose para manter o nível de SNR [14].

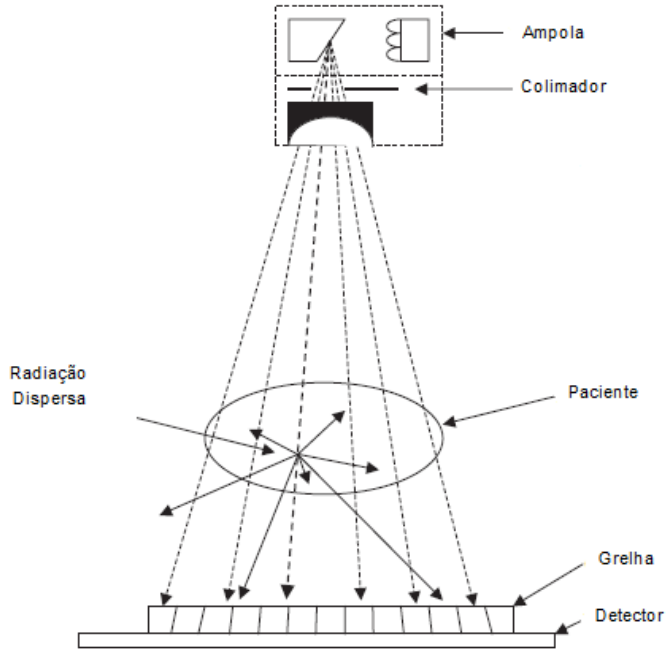

Figura 2.3 Esquema das trajectórias da radiação (normal e espalhada) e vários componentes da produção e detecção de raios-X. Adaptado de [14].

Existem dois métodos de formação de raios-X: radiação característica e Bremsstrahlung (ou de travagem), sendo que ambos, são usados em diagnóstico e radioterapia.

No primeiro, os electrões interagem com os electrões nas orbitais dos átomos do alvo e se o electrão incidente possuir uma energia cinética maior ou igual do que a energia de ligação do electrão, este último pode ser ejectado do átomo. A lacuna deixada por este electrão pode ser depois preenchida por um electrão de camadas superiores, esta transição provoca a produção de um fotão com a energia igual à diferença das energias das camadas. Para um alvo de tungsténio, estas energias são de 69 keV, para a transição para a camada K e de 11 keV, para a transição para a camada L (estas, devido à sua baixa energia não são importantes para aplicações médicas, sendo que a maior parte é atenuada ainda dentro da ampola).

No segundo, o feixe de electrões é travado devido à interacção da força dos seus electrões com o núcleo do alvo. Esta travagem implica a perda de energia cinética por parte dos electrões, através da produção de fotões com energia igual à da energia cinética perdida [14, 15]. O espectro de raios-X é observável na Figura 2.4 e é possível distinguir a radiação característica (origina os picos) e a radiação Bremsstrahlung (zona contínua do espectro).

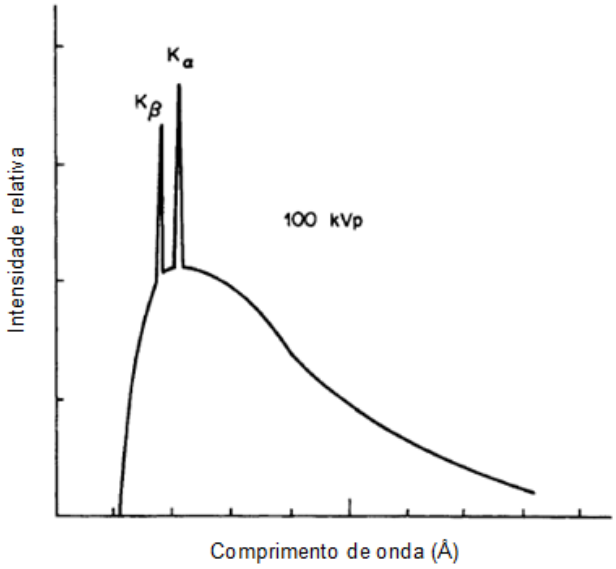

Figura 2.4 Espectro de raios-X. É possível observar os raios-X característicos, K_α e K_β e os raios-X de Bremsstrahlung (contínuo). Adaptado de [10].

Com os avanços tecnológicos, os detectores passaram de simples películas fotográficas a detectores compostos por novos elementos como cintiladores e fotomultiplicadores [11, 12]. Os cintiladores baseiam-se na propriedade que certos materiais têm de emitir luz quando irradiados com radiação ionizante (convertem fotões de raios-X em múltiplos fotões visíveis). Os fotomultiplicadores por sua vez convertem esses fotões em impulsos eléctricos que são depois amplificados [16]. Actualmente os detectores são mais sofisticados, sendo os mesmos descritos na secção referente à mamografia digital.

2.1.2 Interacção dos raios-X com a matéria

Existem vários processos pelos quais os raios-X interagem: dispersão de Rayleigh, efeito fotoeléctrico, efeito de Compton, produção de pares ou tripletos e fotodesintegração, no entanto o estudo detalhado de cada um dos processos, não entra no âmbito desta tese, remetendo-se o mesmo para a bibliografia [14, 15, 17].

Os principais mecanismos[2] de interacção dos raios-X com a matéria são o efeito fotoeléctrico (para baixas energias), o efeito de Compton (para energias intermédias) e a produção de pares (para altas energias) (Figura 2.5).

[2] A dispersão de Rayleigh também ocorre para baixas energias, mas a sua probabilidade de ocorrência é reduzida

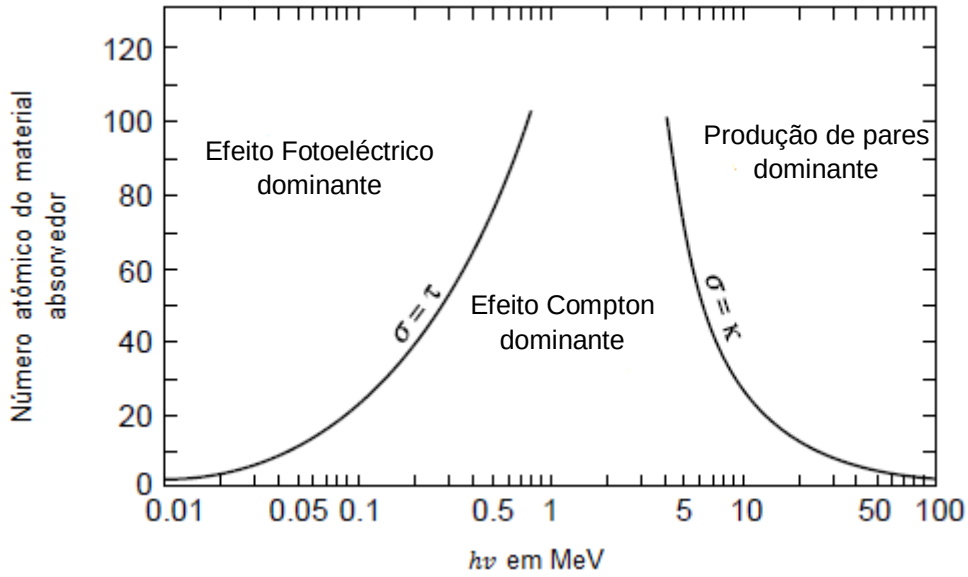

Figura 2.5 Os três mecanismos de interacção principais dos raios-X com a matéria. As curvas representam os valores de Z e *hν* para quais os efeitos são iguais. Adaptado de[14]

Nos exames de mamografia/DBT são usadas energias baixas (da ordem dos 27 keV), pelo que a produção de pares não contribui para a imagem. Convém detalhar alguns pontos importantes para perceber melhor as interacções mais prováveis nos exames destas técnicas:

- Efeito fotoeléctrico

Neste mecanismo o fotão interage com o átomo alvo e ejecta um electrão de uma das suas camadas (K, L, M ou N). O electrão ejectado possui uma energia que é dada pela equação 2.1:

$$E_{cinética} = hv - E_{ligação} \quad \textbf{(2.1)}$$

Sendo hv a energia do fotão incidente e $E_{ligação}$ a energia de ligação do electrão.

A probabilidade do fotão sofrer efeito fotoeléctrico é dependente de Z^3. Este facto traduz-se numa absorção diferenciada dos ossos, músculos e gordura o que providencia um elevado contraste na imagem de raios-X. Em mamografia a imagem é obtida devido à absorção diferenciada entre tecido adiposo, tecido fibroglandular, tecido tumoral e calcificações. Este é o efeito predominante em mamografia devido às baixas energias usadas.

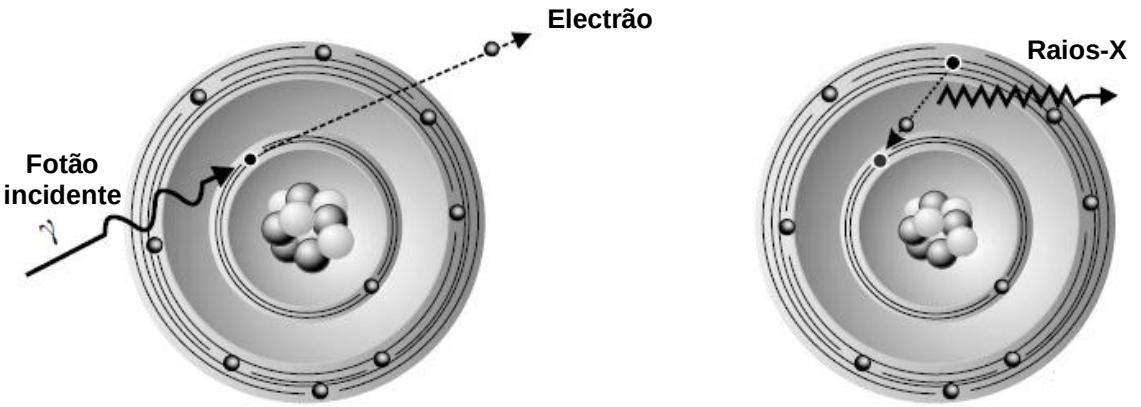

Figura 2.6 Efeito fotoeléctrico. Adaptado de [18].

- Efeito de Compton

No efeito de Compton (ou dispersão de Compton) o fotão (de energia hv_0) transfere parte da sua energia para um electrão livre duma camada exterior, sendo este ejectado com uma energia cinética E e a um ângulo ϕ relativamente à direcção do fotão incidente. O fotão é disperso com um ângulo θ a uma energia cinética mais baixa hv'. Através da conservação da energia e do momento, podem-se tirar as seguintes relações[3]:

$$E = hv_0 \frac{\alpha(1-\cos\theta)}{1+\alpha(1-\cos\theta)} \quad \textbf{(2.2)}$$

$$hv' = \frac{1}{1+\alpha(1-\cos\theta)} \quad \textbf{(2.3)}$$

$$cotg\phi = (1+\alpha)tg\theta/2 \quad \textbf{(2.4)}$$

O efeito de Compton não depende do número atómico mas sim da densidade física do material, este facto faz com que o aumento das energias dos raios-X para a zona de Compton, provoque uma redução de contraste. A imagem obtida através de interacções por efeito de Compton, reflecte diferenças do material, em termos de densidade física, como tal, é mais indicado para o diagnóstico de zonas do corpo, que possuam contraste intrínseco como a cavidade peitoral (constituída por osso, tecidos moles e ar).

[3] $\alpha = \frac{hv}{m_0c^2}$ e $m_0c^2 = 511\ keV$

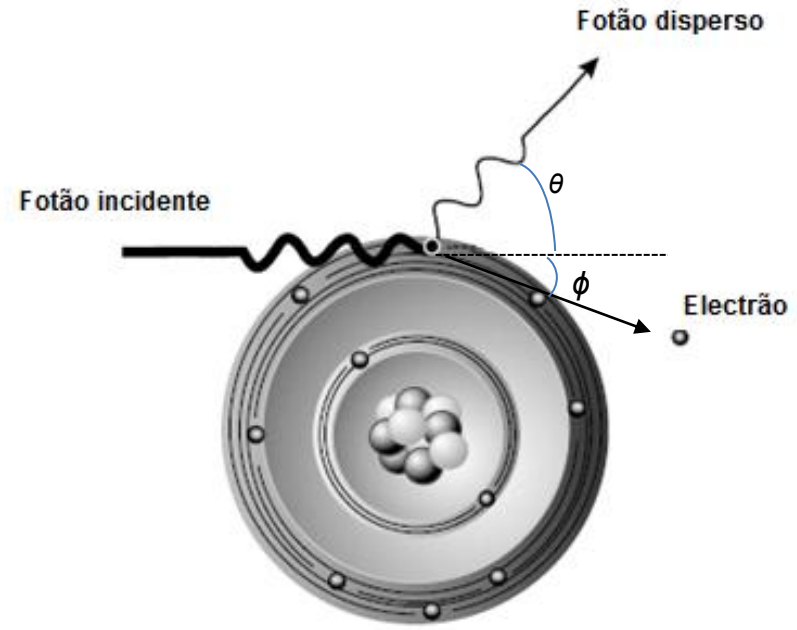

Figura 2.7 Efeito de Compton. Adaptado de [18].

2.1.2.1 Atenuação dos fotões na matéria

Como resultado dos efeitos de interacção referidos atrás, quando um feixe de raios-X atravessa um meio, a sua intensidade diminui, ou seja, existe uma diminuição do número de fotões nesse feixe. Este fenómeno designa-se por atenuação e pode ser representado pela equação 2.5, que representa a razão entre a intensidade de entrada (I_o) e de saída (I) do feixe, ao atravessar uma distância x de um meio atenuador e em que μ representa o coeficiente linear de atenuação (propriedade do meio).

$$\frac{I}{I_o} = e^{-(\mu x)} \qquad \textbf{(2.5)}$$

O coeficiente linear de atenuação é maior para tecidos densos, como o osso e menor para tecidos pouco densos, como os tecidos moles e a gordura e o seu valor depende, geralmente, da energia dos fotões, no número atómico médio e na espessura do material absorvedor: quanto menor a energia dos fotões e quanto maior o número atómico médio ou a espessura, maior a atenuação.

2.2 Mamografia de raios-X

A mamografia de raios-X é a principal técnica usada actualmente para o rastreio do cancro da mama, sendo ideal para observar pequenos tumores e sinais indirectos de cancro [19-22].

2.2.1 Mamografia convencional

Os primeiros aparelhos de mamografia apareceram no final década de 70 [4] e desde então foram realizados vários estudos que demonstraram uma relação directa, entre a introdução dos aparelhos de mamografia com a redução da taxa de mortalidade devido ao cancro da mama: L. Tabar *et al.* [4] conduziram estudos nos quais compararam as mortes de mulheres (com idades entre os 20 e os 69 anos) provocadas pelo cancro da mama, nos 20 anos anteriores e posteriores à introdução dos aparelhos de mamografia tendo concluído que a introdução dos programas de rastreio provocou uma diminuição de 40% a 50% na taxa de mortalidade devido ao cancro da mama. A mamografia convencional é uma técnica ideal para implementar programas de rastreio devido à sua sensibilidade e baixo custo.

Figura 2.8 Aparelho de mamografia Siemens MAMMOMAT 3000 Nova [23].

2.2.2 Desvantagens

Apesar de ser a técnica mais usada actualmente para o diagnóstico do cancro da mama, apresenta no entanto algumas desvantagens: menor capacidade de detecção de tumores em mamas densas [24]; sobreposição de tecidos, devido à imagem obtida ser uma representação 2D do volume tridimensional da mama [25]; o desconforto causado pela compressão da mama durante a realização do exame [26]; o facto de a sensibilidade e

especificidade[4] do diagnóstico serem limitadas pelo observador [27]; a elevada percentagem de falsos positivos: Lindfors *et al*. [28] conduziram um estudo para analisar o stress criado por falsos positivos em mamografia e os resultados mostraram que 40% (46 de 115) das mulheres inquiridas, já tinham tido um falso positivo; Elmore *et al*. [29] estimaram em 49% o risco cumulativo de falsos positivos após 10 exames de mamografia.

Outra das limitações inerentes à mamografia convencional prende-se com o facto de os detectores usados, nomeadamente, as películas fotográficas, servirem como dispositivo para guardar e mostrar a imagem, além da sua função normal de detecção dos raios-X [20, 21, 30]. O melhorar da tecnologia e a introdução de computadores e detectores mais modernos nos aparelhos de diagnóstico, permitiu o desenvolvimento da mamografia digital.

2.2.3 Mamografia Digital

De todas as técnicas de diagnóstico por raios-X, a mamografia foi a última a adoptar a tecnologia digital. Porém, actualmente, a mamografia digital já ultrapassa a mamografia convencional em países como a Alemanha, onde perfaz 88% dos aparelhos instalados [19].

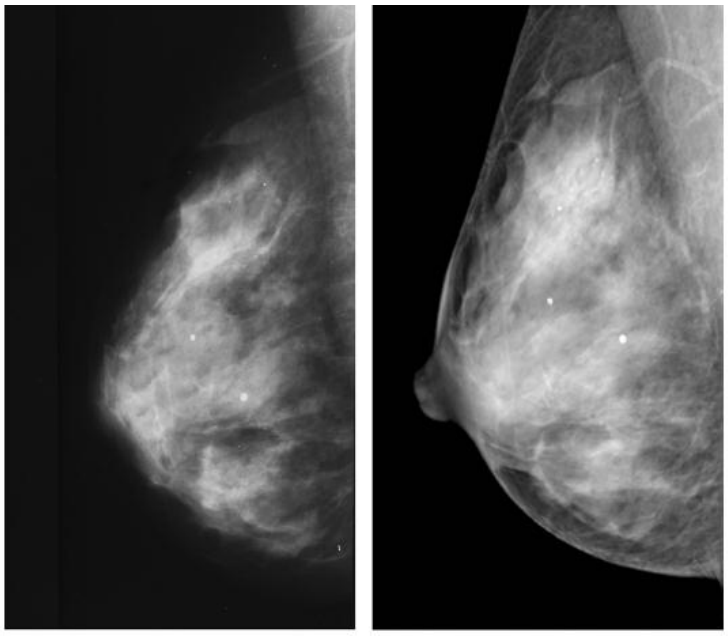

Figura 2.9 Comparação de uma imagem de mamografia convencional (esquerda) com uma imagem de mamografia digital (direita). O paciente é uma mulher de 57 anos de idade, com parênquima mamário e microcalcificações benignas [31].

A principal vantagem da mamografia digital relativamente à mamografia convencional prende-se com o facto de individualizar as funções para as quais os antigos detectores eram desenhados: aquisição, processamento e *display* da imagem. Ao serem desempenhadas independentemente, passou a ser possível optimizar cada uma das partes individualmente, o que levou a melhoramentos na qualidade da imagem (devidos a melhoramentos na aquisição e no processamento), no armazenamento dos exames (com o aumento quase exponencial do

[4] $Especificidade = \frac{Verdadeiros\ Negativos}{Falsos\ Positivos + Verdadeiros\ Negativos}$

tamanho dos discos rígidos passou a ser possível guardar milhões de imagens num só computador) e à introdução de novas ferramentas de manipulação das imagens, nomeadamente ao nível de *software* [19, 21, 22, 30, 31].

Em mamografia digital existem dois tipos de tecnologias: sistemas *offline* e sistemas *online* (mais recentes). Os primeiros usam uma cassete removível como detector usando depois um aparelho de leitura externo para gerar a imagem digital. Mais especificamente, os sistemas *offline* possuem uma placa de fósforo que guarda uma "imagem latente" ou seja, guarda o padrão de absorção dos raios-X. Essa placa é depois inserida num aparelho de leitura externo que irradia a placa com um laser. Ao ser irradiada pelo laser a placa liberta luz proporcionalmente aos raios-X absorvidos. A luz emitida entra depois num circuito de fotomultiplicadores, gerando assim um sinal eléctrico que é depois logaritmicamente amplificado, digitalizado e processado (conversão indirecta). A placa de fósforo é depois irradiada com luz branca de forma a apagar os dados do exame para que possa ser reutilizada. Por outro lado, os sistemas *online*, consistem em aparelhos nos quais o detector é integrado no sistema de mamografia e a imagem é gerada no mesmo aparelho [32]. Os sistemas *online* podem ser de conversão directa ou conversão indirecta.

Com os avanços tecnológicos os detectores sofreram alterações significativas. Os detectores mais recentes realizam directamente a conversão da energia dos fotões de raios-X para sinais eléctricos. Os detectores mais comuns são construídos usando o fósforo como absorvedor dos raios-X. Os novos detectores por outro lado, passaram a ser desenhados a partir da tecnologia de selénio amorfo[5] (a-Se), que permite que a energia dos raios-X seja convertida directamente para corrente eléctrica, ao invés de se converter primeiro essa energia em luz visível e apenas depois em corrente eléctrica, melhorando dessa forma a eficiência da aquisição [31]. O método de funcionamento consiste na conversão da energia dos fotões incidentes, em condutores de cargas livres, pelo fotocondutor a-Se, sendo esses condutores direccionados para um de dois eléctrodos presentes no sistema, através da aplicação de um campo eléctrico. Cada pixel do detector armazena uma certa carga que é depois lida ligando um transístor de película fina (TFT – *thin-film transistor*) [33].

[5] Detector do aparelho de DBT usado neste trabalho

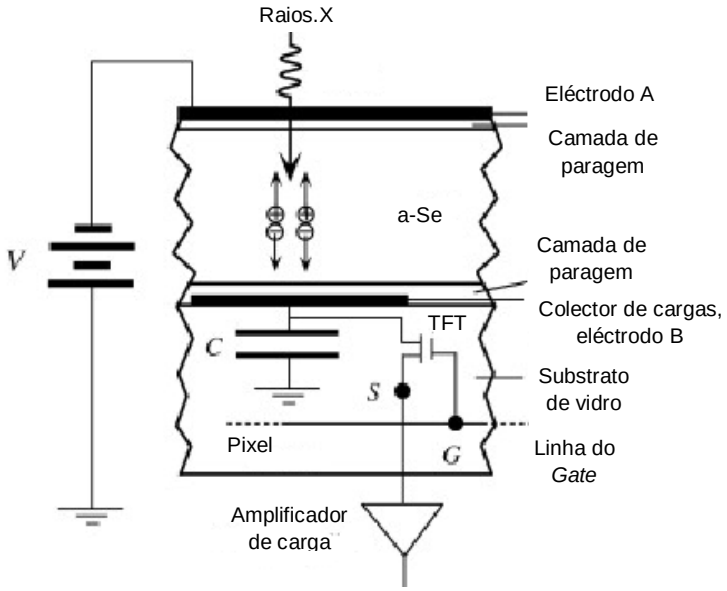

Figura 2.10 Esquema de um corte de um pixel de um detector de a-Se. Adaptado de [33]

Existem ainda algumas técnicas *online* (de conversão indirecta) que não foram referidas como *flat-panels* de silicone amorfo e dispositivos de carga acoplada (CCD – *charge coupled device*), remetendo-se para a bibliografia se se pretender um estudo mais aprofundado das mesmas [14, 19, 31, 32]

A passagem do formato analógico para o digital, permitiu uma nova forma de diagnóstico: diagnóstico assistido por computador (CAD – baseia-se na implementação de algoritmos como redes neuronais), que não sendo um diagnóstico cem por cento fiável, é usado como segunda opinião que permite um aumento de sensibilidade. Esta técnica apresenta um melhor desempenho na detecção de microcalcificações comparativamente a massas [34-37].

2.3 Tomossíntese digital mamária

O princípio da tomossíntese é conhecido desde os anos 30 no entanto, apenas na última década, e graças à evolução dos detectores usados em imagem médica de raios-X, se procedeu à sua aplicação a nível pratico [38].

Apesar das vantagens da mamografia digital relativamente à mamografia convencional, o problema de sobreposição de tecidos, principalmente em mamas densas, manteve-se, uma vez que em imagens bidimensionais há sempre o risco de tecido denso se sobrepor ou subpor a um tumor, ficando este extremamente difícil de discernir num exame radiológico [26, 38-45]. Em caso de dúvida o paciente terá que se submeter a outros métodos adicionais (como US, MRI ou mesmo uma biopsia) para confirmar, ou não, a presença do tumor.

A tomossíntese digital mamária (DBT) é uma técnica bastante recente (a primeira publicação sobre DBT data de 1997 por Niklason *et al*. [46]) que visa ultrapassar, entre outros, o problema da sobreposição dos tecidos em mamas densas, através da aquisição de múltiplas imagens (múltiplos ângulos e pequenas doses de raios-X) da mama que se encontra numa posição estacionária tal como esquematizado na Figura 2.11.

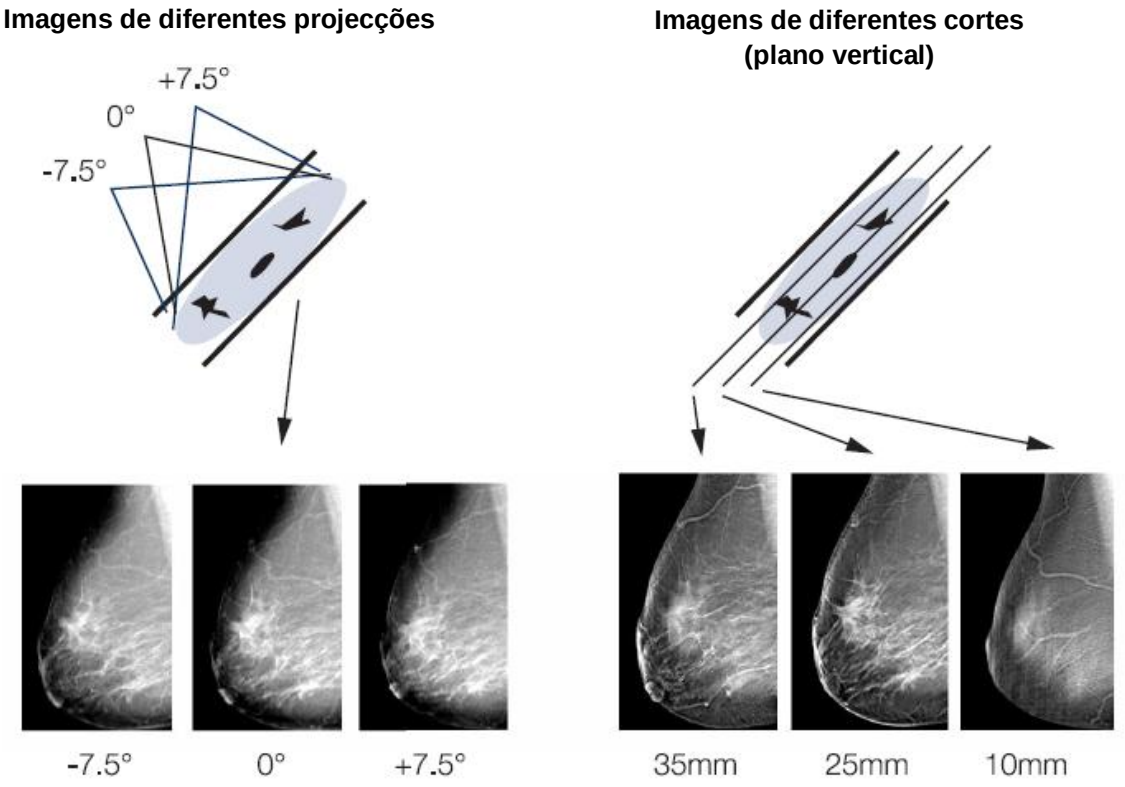

Figura 2.11 Aquisição de imagens em diferentes posições angulares e cortes verticais. Adaptado de [25].

As máquinas de DBT podem consistir apenas numa actualização das máquinas de mamografia digital, se estas possuírem um eixo de rotação próximo do volume a reconstruir [25].

2.3.1 Aquisição das imagens

Após a compressão da mama, a fonte de raios-X é rodada numa amplitude angular limitada, sendo que o número de projecções pode variar. São normalmente adquiridas cerca de 10 a 25 imagens, usando, para o efeito, pequenas doses de raios-X em cada aquisição (cada aquisição corresponde a 5-10% da dose de uma aquisição de mamografia [25]). O intervalo angular entre cada projecção também é variável. Existem dois mecanismos de movimento da ampola de raios-X: contínuo ou *step-and-shoot*. No primeiro, enquanto a ampola se movimenta, são usados pulsos curtos de raios-X para diminuir a desfocagem da imagem devido ao movimento de rotação, enquanto no segundo, a ampola tem que parar em cada posição angular antes de ser ligada.

Os vários parâmetros (amplitude angular, número de projecções e método de aquisição) têm que ser optimizados de forma a obter a imagem desejada, tendo em conta que um maior número de projecções permite maior informação, mas também aumenta o tempo de aquisição, o que por sua vez pode provocar artefactos de movimento do paciente, deteriorando dessa forma a imagem. Os detectores usados são de selénio amorfo que, tal como foi referido no capítulo anterior, convertem directamente a energia dos raios-X em energia eléctrica, facto esse que é essencial, uma vez que as doses de cada aquisição são pequenas e é necessário um rendimento energético bastante elevado (estes detectores possuem uma eficácia quântica de detecção (*Detective Quantum Efficiency*) superior a 95%) [25, 26, 47]. Os detectores também podem ser desenhados de duas formas diferentes: estacionários e móveis.

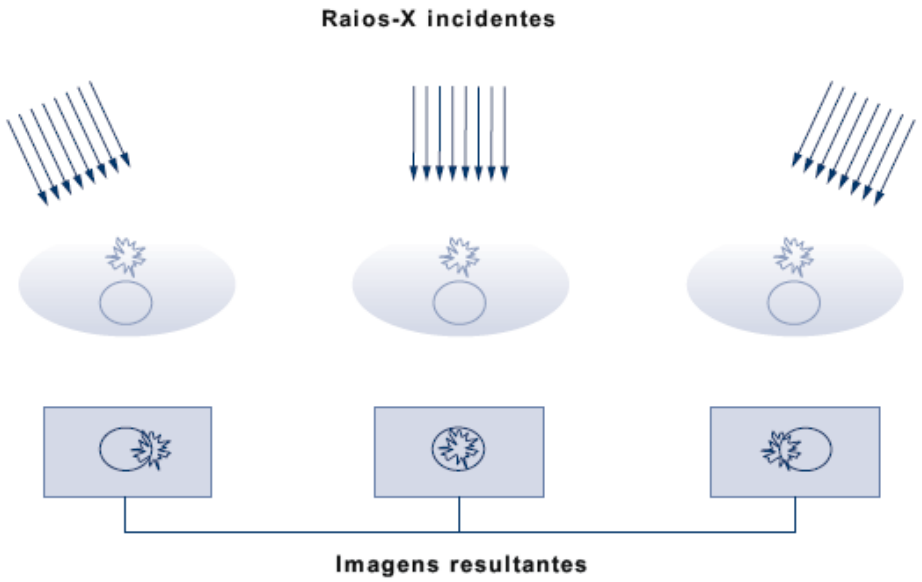

Figura 2.12 Aquisição de imagem, a diferentes ângulos, em DBT. Adaptado de [25]

2.3.2 Vantagens e desvantagens da DBT

Como já foi referido, a principal vantagem da DBT, prende-se com o facto de a aquisição de múltiplas imagens providenciar uma informação tridimensional do objecto, bastando para tal analisar os vários cortes da mama obtidos. Ao obter cortes de espessura reduzida (em geral, 1 mm [25]) elimina-se o problema da sobreposição de tecidos, existente na mamografia convencional (Figura 2.13). Outra vantagem da DBT (apesar de ser uma vantagem indirecta que resulta da eficácia da DBT) prende-se com uma redução da solicitação de novos exames (redução das taxas de *recalls*): o melhor discernimento das lesões traduz-se numa diminuição dos falsos positivos, não sendo, nesses casos, necessário recorrer a novos exames.

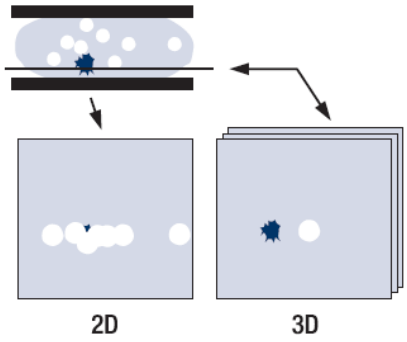

Figura 2.13 Representação esquemática da vantagem da DBT (direita) relativamente à mamografia (esquerda) no respeitante à sobreposição de tecidos [25].

Apesar de ser uma técnica recente, têm sido realizados, ao longo dos últimos anos, alguns estudos que visam analisar as vantagens da DBT em relação à mamografia digital. Vecchio S. *et al.* [44], compararam vários conjuntos de imagens de mamografia digital com imagens de DBT: imagem 2D de mamografia digital *vs* reconstrução 3D de DBT e imagem 2D de mamografia digital *vs* reconstrução do corte central (0º) de DBT, tendo concluído, com o primeiro estudo que as imagens obtidas de DBT apresentam muito mais detalhe do que a imagem obtida por mamografia digital, o que se deve principalmente ao facto de as imagens de mamografia digital apresentarem sobreposição de tecidos, o que torna o diagnóstico bastante menos fiável. A comparação da projecção central de DBT com a imagem de mamografia digital mostrou que a qualidade de imagem é semelhante nas duas técnicas, o que implica que a DBT seja uma técnica mais completa na medida em que o aparelho permite acomodar as duas modalidades de imagem.

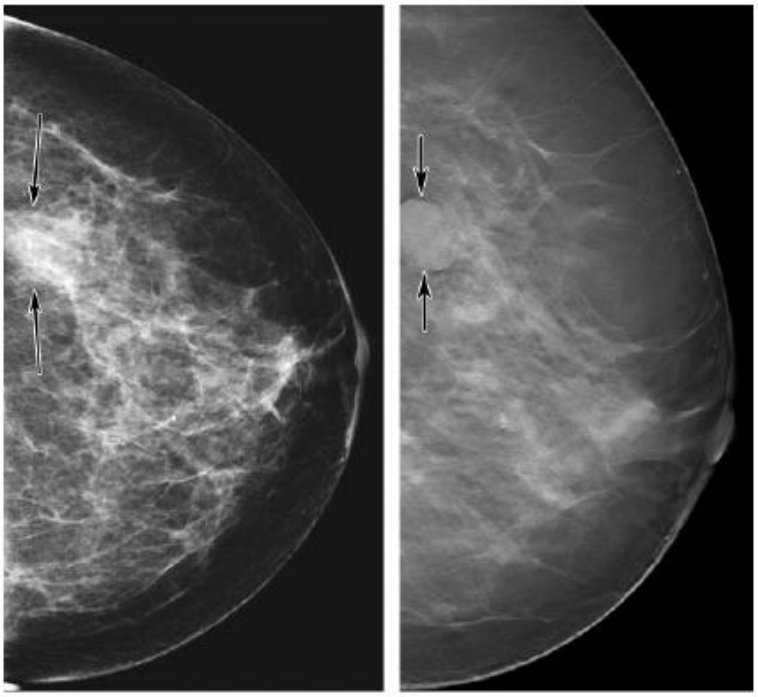

Figura 2.14 Na DBT (direita) é possível analisar com melhor exactidão uma zona suspeita em mamografia (esquerda) [26].

Um estudo realizado por Gur *et al.* [42] comparou a taxa de *recalls* da DBT e da mamografia digital. Para tal, oito radiologistas experientes analisaram 125 exames (35 com sinais de cancro e 90 sem sinais de cancro). O estudo mostrou uma redução de 10% na taxa de *recalls* para a DBT e de 30% para um uso combinado de mamografia digital e DBT, para exames não contendo sinais de cancro. No entanto, os autores apontam que os números em prática clínica serão melhores (as taxas de *recall* deste estudo são, segundo os autores, duas, a duas vezes e meia mais altas do que o esperado do mesmo grupo de radiologistas em prática clínica) devido às condições em que a experiência foi feita: foi pedido aos radiologistas que analisassem os exames como se fosse o inicial e os radiologistas não tinham acesso ao historial clinico dos pacientes. Poplack *et al*. [48], conduziram um estudo, no qual compararam a qualidade de imagem e a taxa de *recalls* da DBT relativamente à mamografia. Para a realização do estudo foram realizados exames de DBT a mulheres com exames inconclusivos de mamografia digital. Os resultados mostraram-se bastante satisfatórios quer a nível de qualidade de imagem, no qual a imagem do exame de DBT foi classificada como igual ou superior em 89% dos casos, e a taxa de *recall* apresentou uma redução de 40% (facto que os autores atribuíram à não sobreposição de tecidos em DBT). No entanto, é referido que a mamografia apresenta um melhor discernimento de microcalcificações.

Outras das vantagens prendem-se com o facto de a DBT, devido ao facto de ser uma modificação da mamografia digital, possuir todas as vantagens que esta possui [26]; redução da dose de forma indirecta (devido à eficácia da DBT e consequente redução da taxa de *recalls,* evita-se expor os pacientes a novos exames e portanto a uma repetida dose) [25, 26]; compressão reduzida da mama, uma vez que a compressão da mama em mamografia digital é bastante elevada de forma a tentar reduzir a sobreposição de tecidos, já em DBT, a

compressão é feita apenas para garantir imobilização e reduzir artefactos de movimento [25, 26].

Nem todos os estudos são conclusivos quanto às vantagens da DBT relativamente à mamografia digital. Teertstra *et al.* [45] realizaram um estudo tendo por base exames de 513 mulheres e reportaram uma sensibilidade de detecção igual em DBT e mamografia digital (92,9%), no entanto os resultados da especificidade da DBT mostraram ser ligeiramente inferiores aos da mamografia digital (84,4% contra 86,1%). Os autores referem que esta diferença pode advir do facto de os radiologistas estarem habituados à análise de mamografia, o que pode interferir na análise de lesões em DBT.

As desvantagens da DBT devem-se principalmente aos artefactos de movimento provocados pela rotação da ampola de raios-X; o aumento do tempo da realização de um exame e ao lado mais técnico, uma vez que é necessária uma nova habituação por parte dos radiologistas devido às novas características apresentadas pela DBT relativamente à mamografia digital e aumenta também o tempo de análise de um exame uma vez que é necessário analisar os vários cortes obtidos [26, 49].

2.3.3 Estado da arte

O primeiro aparelho de DBT a ser aprovado pela Food and Drug Administration, e também o primeiro a estar disponível comercialmente, foi desenvolvido pela Hologic, Inc. tendo recebido aprovação apenas em Fevereiro de 2011 [7, 8].

Figura 2.15 Aparelho de DBT: Selenia® Dimensions®, propriedade da Hologic, Inc [25].

Em Portugal, o único aparelho de tomossíntese existente encontra-se no serviço de imagiologia do Hospital da Luz e é fabricado pela Siemens.

Os estudos que têm sido feitos ao longo dos últimos anos sobre DBT incidem, na sua maioria, nos problemas que advêm do movimento de rotação da fonte de raios-X.

G. Yang *et al.* [49] conduziram estudos num aparelho de DBT com fonte estática ao qual chamaram s-DBT (*Stationary Digital Breast Tomosynthesis*). O objectivo do estudo era diminuir o tempo de aquisição e melhorar algumas desvantagens que advém do movimento da fonte de raios-X e da aquisição prolongada, como por exemplo o *motion blur* (efeito de desfocagem da imagem devido ao movimento). Para tal substituíram o eixo de rotação, presente numa máquina tradicional de DBT, por um tubo emissor de múltiplos feixes de raios-X, que são ligados e desligados com recurso a um sistema electrónico (Figura 2.16). Os resultados preliminares mostraram que a s-DBT é uma opção a ter em conta uma vez que conseguiu diminuir o tempo de aquisição (e consequente diminuição da dose a que o paciente fica sujeito) e melhorou a resolução espacial, relativamente a um aparelho de DBT normal.

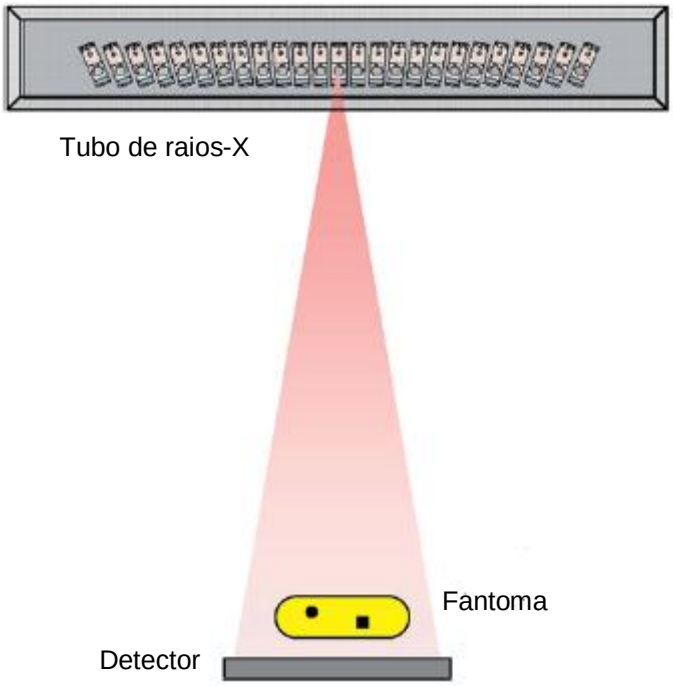

Figura 2.16 Esquema de um aparelho de s-DBT. Adaptado de [49].

Outro estudo teve também como objectivo diminuir o *motion blur* provocado pela rotação da fonte. Zhou W. *et al.* [41, 50] implementaram também uma fonte de múltiplos feixes de raios-X tendo conseguido diminuir o tempo de aquisição das imagens. A qualidade de imagem variou consoante os algoritmos aplicados. Os algoritmos iterativos, nomeadamente o ML-EM e o ART (*algebraic reconstruction technique*) mostraram ser sensíveis a limitações do aparelho construído, ao nível do detector e de parâmetros de medida. No entanto com a aplicação de filtros, as imagens melhoraram substancialmente. Através dos restantes algoritmos aplicados, obtiveram imagens de boa qualidade.

2.4 Reconstrução de Imagem

O output de um exame tomográfico não é uma imagem, mas sim um conjunto de projecções que representam a atenuação dos raios-X (no caso das técnicas tomográficas baseadas em raios-X, como a CT e a DBT) no volume a ser reconstruído, segundo diferentes ângulos. Essas projecções são posteriormente reconstruídas através de algoritmos matemáticos (processos matemáticos passo-a-passo) desenvolvidos para o efeito. O resultado final desta reconstrução consiste numa matriz tridimensional, onde a cada voxel é atribuído um valor proporcional ao coeficiente de atenuação do objecto nesse voxel.

Neste capítulo será feita, inicialmente, uma referência ao conceito de tomografia e serão depois descritos os algoritmos de reconstrução de imagem, no entanto, apenas os algoritmos iterativos, ML-EM e OS-EM, serão tratados com algum detalhe, uma vez que foram os algoritmos usados neste trabalho.

2.4.1 Tomografia

O termo tomografia deriva da palavra grega *tomos*, que significa corte, e caracteriza o processo pelo qual se obtém a imagem 2D de um corte (secção) de um volume 3D. Num exame tomográfico, são obtidas as imagens de vários cortes 2D de um mesmo objecto, sendo assim possível analisar todo o volume do objecto através da análise dos vários cortes obtidos. Um problema simples de tomografia, descrito por G.L Zeng [51], é pensar num parque com duas árvores (Figura 2.17). Se se tirar duas fotografias, de diferentes ângulos, dessas árvores é possível fazer uma planta do parque. Para tal, posicionam-se as fotografias (projecções) nos ângulos a que foram tiradas e traçam-se linhas das árvores, achando-se posteriormente a intersecção entre as linhas das duas vistas.

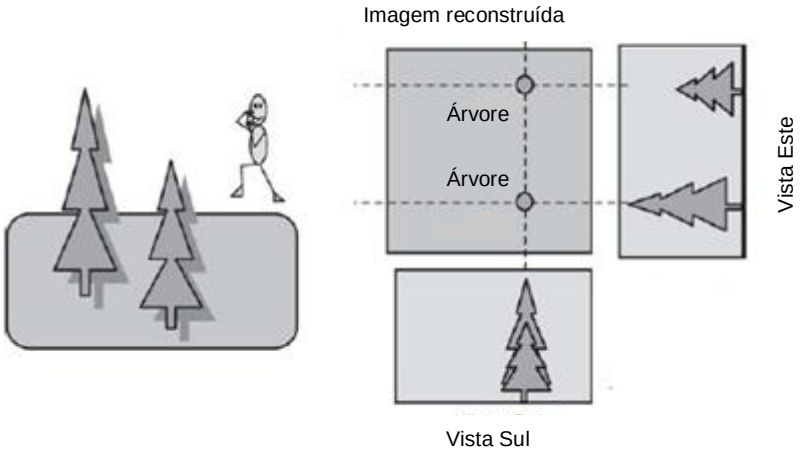

Figura 2.17 Reconstrução do mapa de um parque através de duas vistas. Adaptado de [51].

Matematicamente, o objecto só seria totalmente conhecido, se o número de projecções fosse infinito. Na prática é impossível obter um número infinito de projecções, sendo que a

exactidão da reconstrução está directamente relacionada com o número de projecções adquiridas. Este facto leva a que, em imagem médica, a precisão necessária resulte num número de projecções muito superior ao exemplo representado.

2.4.2 Tomografia Computorizada de raios-X

O objectivo da CT é obter uma imagem dos vários coeficientes de atenuação (μ) dos diferentes materiais do objecto em estudo. O coeficiente de atenuação é uma propriedade intrínseca do material e pode ser calculado, para um determinado voxel j através da equação 2.6.

$$\sum_{j}^{n} \mu_j x_j = \ln(\frac{I_O}{I}) \quad \textbf{(2.6)}$$

Onde μ_j e x_j representam, respectivamente, o coeficiente de atenuação e o comprimento do feixe de raios-X dentro do voxel j e I_o e I representam, respectivamente, os valores das intensidades de entrada e de saída dos raios-X ao passar pelo objecto.

Como se depreende da equação 2.6, com apenas um feixe de raios-X, é impossível determinar os valores de μ para diferentes voxeis. No entanto, com múltiplas projecções é possível obter o coeficiente de atenuação correspondente a cada voxel e ao aplicar níveis de cinzento a diferentes gamas de coeficientes de atenuação é possível representá-los numa imagem tridimensional composta por voxeis com diferentes níveis de cinzento. O resultado de um exame de CT contém uma medida indirecta dos coeficientes de atenuação medidos num determinado *bin* do detector. Essa medida consiste numa gama de números inteiros (-1000 a 3000) designados de unidades de Hounsfield (HU). No caso da DBT, o output dos exames corresponde aos coeficientes de atenuação [13, 52], não sendo necessário calcular os números de Hounsfield para cada voxel.

2.4.3 Algoritmos Iterativos de reconstrução

A reconstrução de imagem através de projecções é, matematicamente, um problema inverso, onde se infere os parâmetros de um sistema partindo dos resultados observados. O problema pode ser descrito, matematicamente, e para técnicas de medicina nuclear, como SPECT e PET, pela equação 2.7

$$Y_i = \sum_{j} a_{ij} f_j \quad \textbf{(2.7)}$$

Em medicina nuclear Y_i corresponde aos dados adquiridos, a_{ij} corresponde aos elementos da matriz de sistema, que no caso destas técnicas, traduzem os processos de detecção e emissão e f_j representa a distribuição de actividade.

Esta equação pode ser adaptada para DBT, onde, e fazendo uma analogia com a equação 2.6, Y_i corresponde a $\ln(\frac{I_0}{I})$, ou seja, é a medida da atenuação dos raios-X emitidos segundo a direcção definida pelo foco e pelo *bin* i do detector (esta direcção é definida como linha de resposta (LOR)); a_{ij} corresponde a x e descreve a probabilidade de os raios-X emitidos segundo uma determinada LOR, terem sido terem sido atenuados num voxel j; f_j corresponde a μ e representa o coeficiente de atenuação no voxel j. Ao conjunto dos elementos a_{ij}, é dada a designação de matriz de sistema. Esta matriz é um modelo geométrico dos processos de transmissão e detecção e a sua implementação depende apenas da posição da fonte de raios-X. A forma como foi desenvolvida a matriz de sistema neste trabalho será descrita com mais detalhe no capítulo seguinte.

Devido ao elevado número de valores a considerar, tanto ao nível das projecções, Y, como dos coeficientes de atenuação, f, este problema não é passível de ser resolvido linearmente.

Para a reconstrução da imagem, através das projecções obtidas no exame, é necessário recorrer a algoritmos de reconstrução. Existem duas grandes classes de algoritmos de reconstrução: analíticos e iterativos, podendo estes últimos ser divididos em algébricos e estatísticos. Esta divisão dos algoritmos iterativos, prende-se com as assunções que se fazem da natureza dos dados. Os algoritmos iterativos estatísticos podem ser ainda classificados em dois grupos, dependendo se assumem uma distribuição de Poisson ou distribuição Gaussiana para o ruído.

Dentro da classe dos algoritmos analíticos destaca-se o *filtered back projection* (FBP). É o algoritmo mais usado em reconstrução de imagem médica [53-60], devido à sua rapidez de execução e facilidade de implementação, no entanto, as imagens obtidas por este algoritmo apresentam artefactos (riscas) [54, 56-58] não sendo também indicado para a reconstrução em técnicas que apresentam um número limitado de projecções, o que acontece em DBT, devido à gama angular limitada a que se realiza o exame [60]. O algoritmo mais usado em DBT (pertencente também à classe dos algoritmos analíticos) é o *shift-and-add,* devido à sua rapidez e simplicidade [61].

Os algoritmos analíticos em geral, pecam pelo facto de ser bastante difícil incorporar modelações dos processos físicos de aquisição de imagem, sem comprometer a vantagem da rapidez e simplicidade e por admitirem que os dados adquiridos são totalmente consistentes com os coeficientes de atenuação no objecto, algo que não acontece devido à existência de ruído [6, 53].

Os algoritmos iterativos de reconstrução foram introduzidos há já largos anos: o ART foi introduzido em 1970 por Gordon *et al.* [62], o ML-EM em 1977 por Dempster *et al.* [63] e o OS-EM em 1994 por Hudson e Larkin [64]. O ART pertence à classe dos algoritmos iterativos algébricos, enquanto o ML-EM e o OS-EM pertencem à classe dos algoritmos iterativos estatísticos. Durante muitos anos os algoritmos iterativos foram preteridos, uma vez que a sua implementação computacional era feita com tempos de processamento bastante superior ao dos algoritmos analíticos. Com a modernização dos computadores e, mais recentemente, com

a emergência do GPGPU, o tempo de processamento destes algoritmos baixou drasticamente [54, 59, 60, 65-71], fazendo com que se levasse à sua implementação devido às suas vantagens relativamente aos algoritmos analíticos, nomeadamente, o facto de ser possível modelar os processos físicos de aquisição e o facto de lidarem muito melhor com o ruído e com dados com baixa amostragem, o que permite uma redução da exposição e do tempo do exame.

Os algoritmos iterativos baseiam-se num procedimento passo-a-passo, que visa estimar, a cada novo passo, os coeficientes de atenuação reais, tendo em conta a estimativa do passo anterior. Estes algoritmos, inicialmente, convergem assimptoticamente e monotonamente para a solução do problema, no entanto, para um número muito grande de iterações a reconstrução é feita com uma maior quantidade de ruído, cabendo assim ao utilizador, decidir quando se deve parar o algoritmo.

De forma a obter resultados satisfatórios, um algoritmo iterativo de reconstrução necessita de ter em conta certos elementos [6, 72]:

i) Um modelo para os dados adquiridos;

ii) Um modelo para representar os coeficientes de atenuação do objecto – $f_j^{(k)}$, valor do coeficiente de atenuação no voxel j, após a iteração k.

iii) Um método para calcular as projecções que seriam detectadas de acordo com uma determinada estimativa do objecto – matriz de sistema composta por elementos a_{ij}, probabilidade de intersecção da LOR i com o voxel j.

iv) Um método para comparar as projecções calculadas com as projecções adquiridas.

v) Um método para comparar as diferenças entre as projecções calculadas e as projecções adquiridas

Os algoritmos são compostos por 4 etapas sequenciais que se repetem em cada iteração: projecção (*forward projection*), comparação, retroprojecção (*backprojection*) e actualização (Figura 2.18).

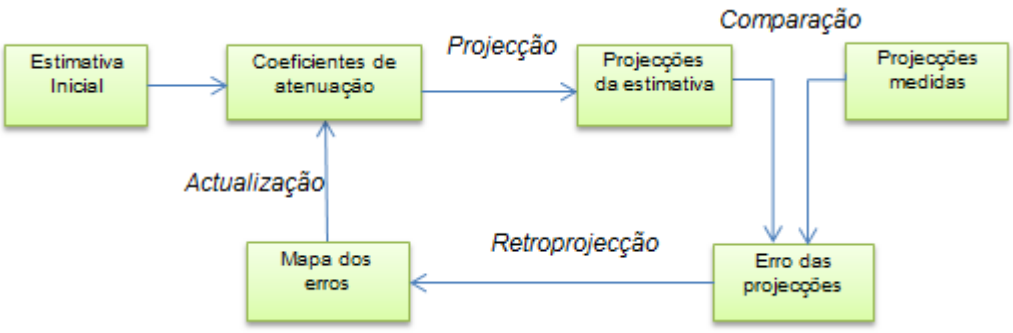

Figura 2.18 Diagrama de fluxo de um algoritmo iterativo de reconstrução

Inicialmente, a estimativa é escolhida pelo utilizador, sendo normalmente uma imagem a zero, no caso de um algoritmo aditivo como o ART, ou no caso de um algoritmo multiplicativo

como o ML-EM e o OS-EM, uma imagem a um. Já com a estimativa inicial, calculam-se as projecções expectáveis caso os coeficientes de atenuação actuais estivessem correctos. Essas projecções são então comparadas com as projecções obtidas, resultando dessa comparação, uma matriz de erro, que é depois retroprojectada, dando origem a um mapa de erros da estimativa actual. A actualização da estimativa é feita através deste mapa de erros, e após actualização, a estimativa é usada para uma nova iteração do algoritmo. Não estando definido um critério de convergência, o número de iterações a realizar, pode ser obtido empiricamente.

2.4.3.1 ML-EM

O algoritmo iterativo de reconstrução de imagem implementado neste trabalho é referente à formulação matemática para tomografia de emissão. Esta implementação deveu-se ao facto de a implementação de CPU com a qual se fez a comparação de resultados, ser também uma implementação para tomografia de emissão.

Como foi referido na secção anterior, o conceito do ML-EM foi introduzido por Dempster *et al.* [63] mas apenas foi aplicado à reconstrução de imagem médica em tomografia de emissão em 1982 por Shepp e Vardi [73] e em 1984 foi aplicado à tomografia por transmissão por Lange e Carsson [74]. No entanto, estes algoritmos têm ganho muito mais popularidade em tomografia de emissão do que transmissão [51, 75].

A estimativa de Máxima Verosimilhança (ML) é um método para estimar os parâmetros de um modelo estatístico. A função de verosimilhança é derivada da estatística de Poisson e é descrita pela equação 2.8.

$$L(f) = \prod_{i=1}^{n} \frac{(Af)_i^{Y_i}}{Y_i!} e^{-(Af)_i} \qquad \textbf{(2.8)}$$

O algoritmo de Máxima Estimativa (EM) é uma técnica iterativa para calcular estimativas de máxima verosimilhança na presença de dados desconhecidos, através de duas importantes etapas em cada iteração: a etapa de expectativa (E-step) e a etapa de maximização (M-step). No primeiro passo (expectativa) são estimados novos coeficientes de atenuação tendo em conta as projecções medidas e os coeficientes actuais, enquanto que no segundo passo (maximização) a função de verosimilhança é maximizada.

Seja λ uma estimativa obtida a partir dos dados y, o critério que procura uma estimativa da imagem (ML) é o que maximiza a probabilidade $P(\lambda|y)$. Para achar a estimativa λ introduz-se, normalmente, o logaritmo da função de verosimilhança como descrito na equação 2.9.

$$L(\lambda) = \ln P(\lambda|y) \qquad \textbf{(2.9)}$$

Uma vez que a função logaritmo é estritamente crescente, o valor de λ que maximiza $P(\lambda|y)$ também irá maximizar $L(\lambda)$. O algoritmo EM tem então como objectivo maximizar $L(\lambda)$.

A fórmula geral do ML-EM é dada por,

$$\lambda_j^{n+1} = \frac{\lambda_j^n}{\sum_i a_{ij}} \sum_i \frac{a_{ij} Y_i}{\sum_t a_{it} \lambda_t^n} \quad \textbf{(2.10)}$$

O objectivo deste algoritmo é descobrir a distribuição de actividade λ (no caso de CT/DBT é descobrir os coeficientes de atenuação) que tem a maior probabilidade de terem originado as projecções medidas Y_i, usando uma matriz de sistema composta por elementos a_{ij}. Calculando o somatório $\sum_t a_{it} \lambda_t^n$, obtêm-se as projecções que seriam medidas caso a estimativa λ_t^n estivesse correcta – etapa de *Projecção* da Figura 2.18. As projecções obtidas são depois comparadas com as projecções efectivamente medidas (Y_i), dividindo as últimas pelo resultado do somatório. Os resultados desta comparação são somados (ponderados pelos elementos da matriz de sistema) obtendo-se assim o factor de actualização. Por fim o factor de actualização é multiplicado pela estimativa anterior e dividido pelo factor de normalização $\sum_i a_{ij}$ – etapa de *Actualização* da Figura 2.18.

Este algoritmo apresenta duas grandes desvantagens: baixa velocidade de convergência e instabilidade [6, 76]. Esta instabilidade deve-se ao facto de o algoritmo procurar iterativamente uma estimativa que melhor caracteriza os dados adquiridos: se os dados adquiridos apresentarem um baixo nível de ruído o algoritmo não origina problemas no entanto, caso os dados apresentem um nível de ruído significativo, a estimativa para qual o algoritmo irá convergir, será também ruidosa.

Lange e Carsson [74] também apresentaram uma formulação para tomografia de transmissão, no entanto salientam que a sua formulação matemática é bastante mais difícil, não tendo conseguido especificar uma solução analítica exacta. A solução aproximada que propuseram é descrita pela equação 2.11.

$$\mu_k^{n+1} = \frac{\sum_{i \in J_k} (M_{ik} - N_{ik})}{\frac{1}{2} \sum_{i \in J_k} (M_{ik} + N_{ik}) l_{ik}} \quad \textbf{(2.11)}$$

Nesta equação M_{ik} e N_{ik}, designam, respectivamente, a estimativa do número de fotões que entram e saem do voxel k para uma determinada LOR i e l_{ik} o comprimento da intersecção da LOR i com o voxel k. Devido ao facto de não ter sido esta a equação usada na realização do trabalho, remete-se o leitor para o trabalho de Lange e Carsson [74], se se pretender um olhar mais detalhado sobre o assunto.

No entanto como foi referido, a utilização deste algoritmo para tomografia de transmissão não teve os mesmos índices de popularidade do que para tomografia de emissão.

2.4.3.2 OS-EM

Devido às limitações do ML-EM descritas na secção anterior, nomeadamente o tempo de convergência, Hudson e Larkin [64] propuseram, em 1994, um algoritmo cujo método se

baseia em dividir as LORs em subconjuntos denominados *subsets* – *Ordered Subsets Expectation Maximization*. A cada subconjunto escolhido é aplicado o ML-EM no final de cada sub-iteração. Uma iteração do OS-EM estará completa, depois de se ter percorrido todos os subconjuntos, no entanto, a imagem terá sido actualizada tantas vezes quanto o número de subconjuntos que tenham sido escolhidos. Ou seja, se se dividir o conjunto das LORs em n subconjuntos, após uma iteração completa do OS-EM, a imagem terá sido actualizada n vezes, comparativamente ao ML-EM no qual a imagem teria sido actualizada uma vez. Quando o número de subconjuntos escolhido é igual à unidade, o OS-EM torna-se igual ao ML-EM. Apesar de o tempo que demora uma iteração do OS-EM poder ser ligeiramente superior ao de uma iteração do ML-EM, a imagem é actualizada n vezes, o que permite obter uma maior velocidade de convergência. As projecções devem ser escolhidas de forma a maximizar a diferença angular entre cada subiteração de modo a obter informações o mais diferente possível em cada subiteração.

Apesar dos bons resultados obtidos com o uso do OS-EM, a sua convergência ainda não foi provada matematicamente e o problema da sensibilidade ao ruído também não é eliminado. O facto de no OS-EM a imagem ser actualizada n vezes, faz com que, neste algoritmo, o ruído comece a aumentar mais cedo [6].

A fórmula do OS-EM é bastante similar á do ML-EM, no entanto, os somatórios referentes às LORs, são substituídos por somatórios das LORs referentes a um determinado subconjunto S_n:

$$\lambda_j^{n+1} = \frac{\lambda_j^n}{\sum_{i \in S_n} a_{ij}} \sum_{i \in S_n} \frac{a_{ij} Y_i}{\sum_t a_{it} \lambda_t^n} \qquad \textbf{(2.12)}$$

2.5 GPU e programação em paralelo

2.5.1 Uma mudança de paradigma

A indústria dos processadores de computadores (CPU), tem tido uma evolução bastante acentuada ao longo das últimas décadas firmemente baseada na Lei de Moore, co-fundador da Intel®, que em 1965, afirmou que o número de transístores que se podem colocar num chip, dobra a cada ano [77]. Se até há bem poucos anos, cada novo modelo de CPU representava um aumento da frequência do relógio da CPU da ordem de 1 GHz relativamente à geração anterior, a tendência recente, tem sido o aumento de núcleos (*cores*) de processamento. Os principais fabricantes de CPUs, a Intel® e a AMD® permitem ao utilizador lidar com giga operações de vírgula flutuante por segundo (GFLOPS – *Giga Floating-Point Operations Per Second*).

Com o melhoramento das CPUs o desempenho das aplicações foi igualmente melhorando. A partir de 2003, o aumento da frequência do relógio da CPU, foi sendo limitado, devido a problemas de consumo de energia e de dissipação de calor. Este facto levou a que os fabricantes de CPUs começassem a desenhar unidades com múltiplos processadores embutidos, ou seja, com vários núcleos de processamento [78]. Este avanço tecnológico trouxe consigo uma mudança de paradigma: se até aqui o *software* corria mais rápido em cada nova geração de CPUs, com a introdução de unidades de vários núcleos de processamento, e o facto de as antigas aplicações poderem apenas correr num dos núcleos de processamento (programação sequencial), as aplicações passaram a ter que ser redesenhadas para aproveitar os vários núcleos. Passou a ser preciso desenvolver programas em paralelo, nos quais múltiplas *threads*[6] de execução cooperam para aumentar a velocidade de processamento. No futuro, a enfâse estará cada vez mais em apostar num aumento do número de núcleos, ao invés de se apostar no aumento de desempenho de cada núcleo [79].

2.5.2 Vantagens da programação em GPU

Desde a introdução dos chips com vários núcleos, passaram a existir duas correntes, na indústria dos semicondutores: *multicore* e *many-core.* A primeira foca-se em manter o desempenho dos programas sequenciais ao mesmo tempo que aumenta o número de núcleos, enquanto a segunda se foca no *throughput* (taxa de transferência de dados) de execução de programas paralelos. Comparando dois representantes de cada uma das correntes é fácil perceber o porquê dos seus diferentes objectivos: um processador topo de gama (*multicore*) Intel® Core™ i7-990X Extreme Edition (2011), possui 6 núcleos enquanto que uma GPU de gama média (*manycore*) NVIDIA® Quadro FX 3800 (2009), correspondente à GPU usada neste trabalho, possui 192 núcleos[7]. Na Figura 2.19 é possível observar a diferença de desempenho entre as duas correntes. O facto das GPUs mais recentes atingirem

[6] Uma *thread* é a menor unidade de processamento que pode ser agendada por um sistema operativo
[7] Convém referir no entanto, que os núcleos de CPU e GPU não são exactamente iguais

desempenhos na ordem dos TFLOPS (*Tera Floating-Point Operations Per Second*) aliado ao seu baixo custo e ao reduzido consumo energético, tornam-na o candidato ideal para cálculos aritméticos intensivos de problemas paralelizáveis [78, 80, 81] .

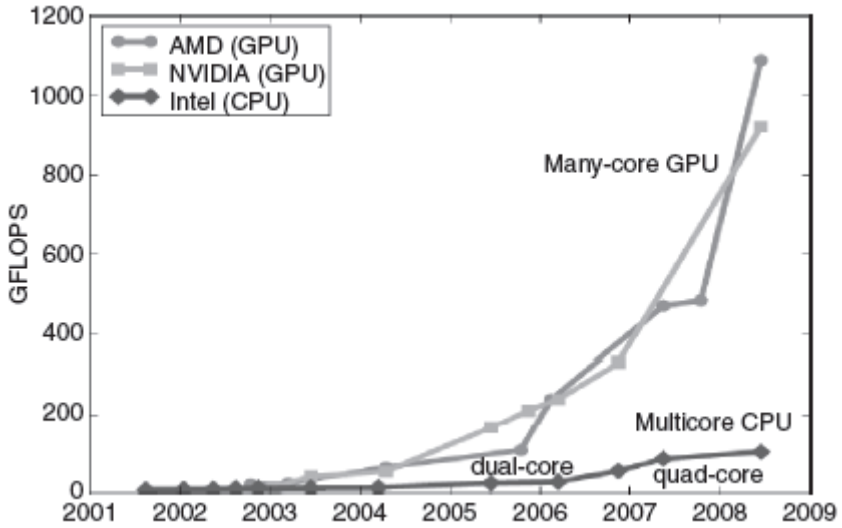

Figura 2.19 Comparação de desempenho entre GPUs e CPUs [78].

Esta diferença de desempenho pode ser explicada tendo em conta a diferente arquitectura de ambos [78, 81]. A CPU é desenhada para processar várias tarefas arbitrárias, como transferência de dados, processamento, agendamento de operações entre outros. Essas tarefas fazem uso de lógica de controlo e de memórias *cache* que permitem reduzir a latência no acesso de dados. A perda de desempenho relativamente às GPUs prende-se com o facto de a lógica de controlo e a memória *cache* não contribuírem para a velocidade de cálculo.

A GPU, por sua vez, é concebida tendo em conta a indústria dos videojogos, que exige um número massivo de cálculos aritméticos por cada *frame* de vídeo, fazendo com que os fabricantes aproveitem ao máximo a área do chip para operações *de* vírgula flutuante.

Convém notar que, por vezes, nem todos os segmentos de um programa são passiveis de ser paralelizados, tendo que se recorrer a programação sequencial juntamente com programação paralela num mesmo programa. Este facto traduz-se num limite de redução do tempo de execução de um programa (*speedup*) e pode ser expresso pela lei de Amdahl [82] da seguinte forma

$$Speedup = \frac{1}{(1-P) + \frac{P}{N}} \quad \textbf{(2.13)}$$

Onde P é a porção de algoritmo que é paralelizável e N, o número de processadores que corre essa mesma porção do código. Assumindo um número de processadores elevado, ou seja, transformando a equação em:

$$Speedup = \frac{1}{1 - P} \quad \textbf{(2.14)}$$

É fácil verificar, que quanto maior a porção de código paralelizável, maior o *speedup* que se obtém. Existem problemas em que não se justifica o uso da programação em paralelo, como por exemplo, problemas que envolvam um reduzido número de dados.

Alguns estudos comprovam a eficiência da GPU na implementação de algoritmos iterativos de reconstrução: Chidlow e Möller [68] implementaram versões modificadas do ML-EM e do OS-EM para tomografia de emissão, com *speedups* de 8.7 (5.5)[8] e 4.2 (5.4) vezes, respectivamente; também para tomografia de emissão Pratx *et al.* [71] implementaram um OS-EM modificado (OS-EM *list-mode*) com *speedup* de 51 vezes e sem diferenças relativamente à qualidade da imagem reconstruída em GPU *vs* CPU; Xu e Mueller [69] implementaram algoritmos analíticos e iterativos em GPU, para tomografia de transmissão, e obtiveram *speedups* de uma ordem de magnitude em ambos; Vintache et al. [65] implementaram um algoritmo iterativo (OSC – *Ordered Subsets Convex*) para reconstrução de imagem em CT com recurso a CUDA™ e obtiveram imagens de qualidade semelhante a CPU e *speedups* de 10 vezes.

2.5.3 CUDA™

Até por volta do ano de 2006, o uso das GPUs em áreas como a reconstrução de imagem médica, finanças, biologia, entre outros, era bastante restrito, uma vez que a sua programação se fazia com recurso a interfaces de programação de aplicativos (API) gráficas. Para programar as GPUs era necessário possuir conhecimentos de OpenGL® e Direct3D®, algo que não está ao alcance do comum investigador científico.

Com o aparecimento do CUDA™ (*Compute Unified Device Architecture*), propriedade da NVIDIA®, o estado da programação em paralelo sofreu grandes avanços. A NVIDIA® dedicou-se ao desenho de uma interface que permitisse uma maior facilidade de programação, sem ser necessário recorrer a APIs gráficas. O CUDA™ permite o uso de linguagens familiares como C e C++ (CUDA C) para aceder às capacidades de programação em paralelo de uma GPU.

2.5.3.1 Modelo de programação em CUDA™

Um programa em CUDA™ pode combinar código sequencial (executado no *host* (CPU)) e código paralelo (executado no *device* (que neste caso é uma GPU, mas pode ser outro equipamento de processamento em paralelo)). O código paralelo é desenvolvido em ANSI C com extensões próprias para lidar com as funções que tratam de dados em paralelo.

[8] Os valores dentro de parênteses reflectem o *speedup* obtido numa versão alternativa de implementação

Estas porções de código que tratam os dados em paralelo são designadas de *kernels* e são chamadas a partir do programa sequencial (Figura 2.20).

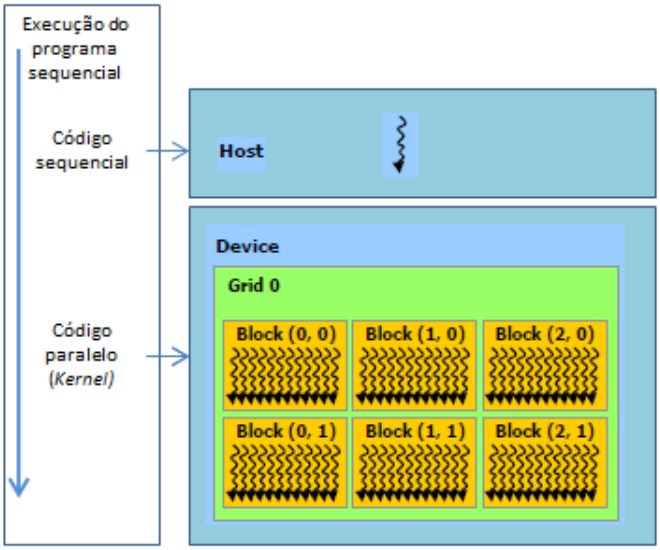

Figura 2.20 Diagrama de um programa em CUDA™. Adaptado de [83].

Quando um *kernel* é chamado a partir do código principal, o mesmo trata de gerar uma grande quantidade de *threads* de forma a aproveitar o paralelismo dos dados (arquitectura de instrução única, múltiplos dados (SIMD – *single instruction, multiple data*)). Estas *threads* são muito mais leves do que as *threads* de uma CPU. O conjunto de todas as *threads* geradas designa-se por grelha. Essa grelha por sua vez é composta por blocos de *threads*. Um exemplo fácil para demonstrar o funcionamento de um *kernel* é por exemplo a adição de uma constante a uma matriz de 1.000x1.000. Neste caso, o *kernel* tem que gerar 1.000.000 *threads* e cada uma será responsável por adicionar a constante a um elemento da matriz. A arquitectura CUDA™ foi desenvolvida tendo por base um conjunto de multiprocessadores de *streaming* (SM) sendo eles os responsáveis pela criação, gestão e execução das *threads*. A execução das *threads* é feita em grupos de 32 *threads*, designados por *warps*.

O CUDA™ permite uma gestão da memória quer do *host* quer do *device*, fazendo com que o utilizador tenha que transferir dados da memória do *host* para a memória do *device* e vice versa. É necessário gerir com algum cuidado estas transferências de memória, uma vez que são feitas com elevada latência [78, 83].

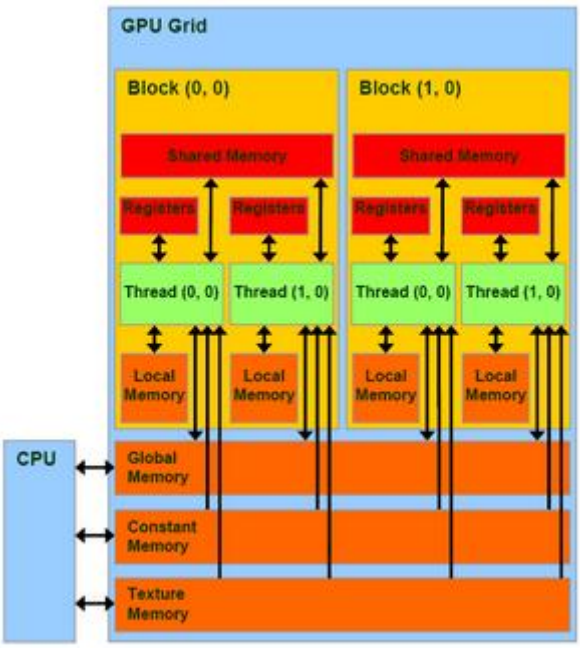

Figura 2.21 Modelo de memórias em CUDA™ [78].

Na Figura 2.21 é apresentado um esquema do modelo de memórias em CUDA™. Da sua análise é possível perceber que o *host* pode transferir dados para/de as memórias global, constante e textura (não usada neste trabalho). Dessas, apenas a memória global permite leitura e escrita por parte das *threads*, sendo as outras duas reservadas apenas para leitura de dados[9]. As *threads* possuem uma memória de rápido acesso, designada por registos, que é de acesso individual por cada *thread*, no entanto a sua capacidade é bastante reduzida. Existe ainda uma memória bastante importante designada por memória partilhada (*shared*) cujo acesso também é feito com reduzida latência e é partilhada por todas as *threads* de um mesmo bloco.

Os aspectos de CUDA™ aqui discutidos evidenciam uma significativa diferença na programação em paralelo, nomeadamente, a diferença de abordagem de um problema, uma vez que é necessário decidir que porções de código são passiveis de ser paralelizadas e de que forma será feita essa paralelização. Esta abordagem inicial de um programa, influencia por si só, a performance de um programa desenvolvido em CUDA™.

[9] As diferenças entre a memória de textura e a memória constante são de índole informática bastante técnica

Capítulo 3

Materiais e Métodos

3.1 Material

3.1.1 Aparelho de DBT

O exame usado neste trabalho foi realizado por um aparelho de DBT da Siemens (*MAMMOMAT Inspiration*), instalado na secção de imagiologia do Hospital da Luz (Figura 3.1). Em cada exame são realizadas 25 projecções numa gama angular de -25º a +25º. Para a avaliação quantitativa das imagens obtidas foi também usado uma variante do fantoma (Gammex 156) específico para controlo de qualidade em mamografia [84].

Figura 3.1 Siemens MAMMOMAT Inspiration

Os componentes do aparelho e as dimensões dos mesmos estão esquematizados na Figura 3.2.

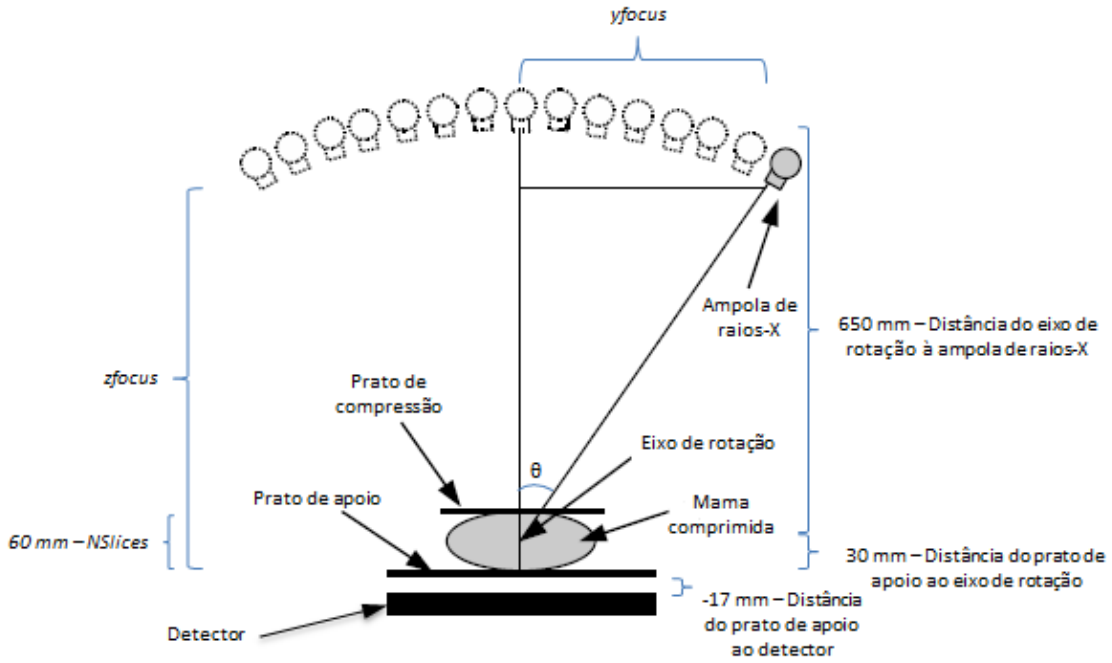

Figura 3.2 Esquema do aparelho de DBT usado neste trabalho. Adaptado de [85].

O detector é constituído por 2816 x 3584 *bins*, em que o tamanho de cada *bin* é de 0.085 mm.

O campo de visão (FOV – field of view) é definido como o volume irradiado pela fonte de raios-X. A sua dimensão é dada por $Detector_{Dim_x} * Detector_{Dim_y} * NSlices = 2816 * 3584 * 60$. Devido a limitações de memória, as dimensões utilizadas foram significativamente inferiores, no entanto a forma como se ultrapassaram as limitações de memória será descrita mais à frente.

3.1.2 *Hardware e Software*

O código foi implementado e testado num computador Intel® Xeon® E5530 2,4 GHz com 12 GB de RAM e a correr o sistema operativo Red Hat Enterprise Linux™ 5.3. A GPU usada foi uma NVIDIA® Quadro® FX 3800 com 1 GB de memória DDR3 e 192 núcleos de CUDA™.

Foi usada a versão 3.2 do CUDA™ Toolkit [86] e a versão 1.3.0 da biblioteca Thrust (biblioteca de CUDA™) [87].

A visualização das imagens em modo *slide-show* foi feita com recurso ao *software* QuasiManager[10].

[10] *Software* desenvolvido internamente pelo Instituto de Biofísica e Engenharia Biomédica

3.2 Métodos

3.2.1 Tópicos fundamentais de programação em CUDA™

Como foi referido na secção anterior, quando é invocado um *kernel* num programa de CUDA™, é gerado um conjunto de *threads* para processar um conjunto de dados em paralelo. Estas *threads* estão dispostas hierarquicamente: uma grelha unidimensional ou bidimensional é composta por blocos de *threads* unidimensionais, bidimensionais ou tridimensionais. Esta organização é definida pelo utilizador aquando da invocação do *kernel* da seguinte forma:

```
Nome_kernel<<<Numero_Blocos,Numero_Threads>>>(argumento1, argumento2);
```

A cada bloco e a cada *thread* gerados, é atribuído um índex único através das variáveis *blockIdx* (variável com dois campos (x,y)) e *threadIdx* (variável com três campos (x,y,z)) respectivamente. Outra variável útil na gestão de *threads* é *blockDim/gridDim* que devolve o tamanho de um bloco/grelha. Existem limites no número de *threads* por bloco e no número de blocos por grelha, devido ao facto de todas as *threads* de um bloco terem que residir no mesmo SM, ou seja, têm que partilhar os recursos de memória de um SM. Assim, enquanto o número máximo de *threads* por bloco é de 512 *threads*, as dimensões máximas da grelha são de 65536 x 65536 blocos. As *threads* de um mesmo bloco podem cooperar umas com as outras a partir da memória partilhada, que é comum a todas as *threads* de um bloco, e a sua execução pode ser sincronizada, de forma a gerir os acessos à memória, através da função *__syncthreads().*

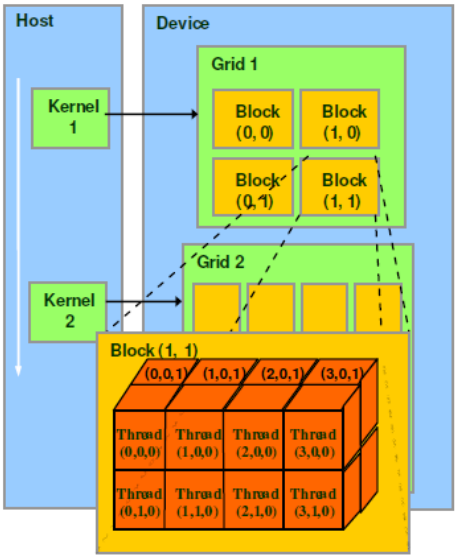

Figura 3.3 Organização hierárquica de *threads* [88].

Relativamente à gestão de memória e transferência de dados, o CUDA™ possui funções próprias para lidar com essas tarefas. Analogamente às funções *malloc()* e *free()* da

linguagem C, existem as funções *cudaMalloc() e cudaFree()* que tratam de reservar espaço na memória global na GPU e de o libertar, respectivamente. A função que trata de transferir os dados do host para o device, e vice-versa, é a *cudaMemcpy()*, na qual é preciso indicar os ponteiros de origem e de destino, a quantidade de bytes a copiar e o tipo de transferência (*host to device* ou *device to host).*

Para melhor compreensão dos conceitos fundamentais de CUDA discutidos previamente, é apresentado de seguida, um programa em CUDA bastante simples, que executa a adição de dois vectores:

```
__global__ void Adicao_vectores(float* vec1, float* vec2,float* vec_output, int dim){

        int i = threadIdx.x + blockDim.x*blockIdx.x;
        if(i < dim)
        vec_output[i] = vec1[i] + vec2[i];
}

int main(){


//Alocação de memória no CPU e inicialização dos vectores
int dim = 1024;
float*vec1 = (float*)malloc(dim*sizeof(float));
float*vec2 = (float*)malloc(dim*sizeof(float));
float*vec_output = (float*)malloc(dim*sizeof(float));

for(int i=0; i < dim; i++){
        vec1[i]=i;
        vec2[i]=i;
        vec_output[i]=0;
}

//Alocação de memória no GPU:
float *vec1_d, *vec2_d, *vec_output_d;
cudaMalloc((void**)&vec1_d, dim * sizeof(float));
cudaMalloc((void**)&vec2_d, dim * sizeof(float));
cudaMalloc((void**)&vec_output_d, dim * sizeof(float));

//Transferência de dados do CPU para o GPU
cudaMemcpy(vec1_d, vec1, dim * sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(vec2_d, vec2, dim * sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(vec_output_d, vec_output, dim * sizeof(float), cudaMemcpyHostToDevice);


//Chamada do kernel de adição de vectores (2 blocos de 512 threads = 1024 threads – uma por cada elemento dos vectores)
Adicao_vectores<<<2,512>>>(vec1_d,vec2_d,vec_output_d, dim);

//Transferência do vector que contém a soma, do GPU para o CPU
cudaMemcpy(vec_output, vec_output_d, dim * sizeof(float), cudaMemcpyDeviceToHost);

//Libertação da memória alocada no GPU
cudaFree(vec1_d);
cudaFree(vec2_d);
cudaFree(vec_output_d);
}
```

3.2.2 Biblioteca Thrust

Numa fase inicial do trabalho, que consistiu numa familiarização com o CUDA™, foram testadas duas bibliotecas de CUDA™, nomeadamente, a biblioteca CUBLAS (que é uma implementação de uma API de álgebra linear BLAS (*Basic Linear Algebra Subprograms*) em CUDA™) cujo objectivo se centra em operações básicas de álgebra linear como multiplicação

de vectores e matrizes e a biblioteca Thrust, que tem como objectivo aproximar a programação em CUDA™ da programação sequencial, usando para tal uma interface semelhante à biblioteca STL (*Standard Template Library*) da linguagem C++.

Para a realização deste trabalho, a biblioteca CUBLAS foi preterida em favor da biblioteca Thrust, uma vez que esta possui uma quantidade bastante superior de funções que foram essenciais à realização do trabalho.

A implementação de certos algoritmos em paralelo pode tornar-se bastante mais complicada do que o seu homólogo sequencial, como são o caso de algoritmos de *sort* (ordenação), de *stream compaction* (compactação de fluxo – eliminação de valores de um *array* que satisfazem, ou não, uma certa condição), de redução de valores (por exemplo o somatório de todos os elementos de um vector em paralelo é uma operação de redução), de avaliação de polinómios, entre outros. Blelloch [89] descreve a operação de somatório de todos os prefixos (do inglês *all-prefix-sums operations*) que permite a implementação dos algoritmos descritos anteriormente em paralelo e define-as como:

"Uma operação de somatório de todos os prefixos tem como *input* um operador binário $\oplus$ com identidade i e um conjunto ordenado $[a_0, a_1, \dots, a_{n-1}]$ de n elementos e devolve o conjunto ordenado $[i, a_0, (a_0 \oplus a_1), \dots, (a_0 \oplus a_1 \oplus \dots \oplus a_{n-1})]$. Por exemplo, se o operador binário $\oplus$ for a soma e se este for aplicado ao conjunto ordenado $[3\ 1\ 7\ 0\ 4\ 1\ 6\ 3]$, a operação devolve o conjunto ordenado $[3\ 4\ 11\ 11\ 15\ 16\ 22\ 25]$". Para um estudo mais detalhado sobre este tema remete-se o leitor para a bibliografia [89-91].

A biblioteca Thrust possui um conjunto bastante alargado de funções que foram de vital importância na realização deste trabalho: ordenação (bastante útil na reordenação da matriz de sistema antes e durante a implementação do OS-EM); compactação de fluxo (usado para eliminação de pontos que se encontravam fora da FOV e para eliminação de zeros – dado que não é possível alocar dinamicamente memória dentro de um *kernel* de CUDA™, é preciso alocar na GPU (antes da chamada do *kernel*) o máximo de memória a usar durante a execução do mesmo, pelo que, muitas vezes durante o programa, se preencheram com zeros posições de vectores e matrizes que nunca chegaram a ser modificadas); redução de valores (essencial para a normalização da matriz de sistema e para os somatórios do algoritmo de reconstrução de imagem – a biblioteca Thrust possui duas versões da operação de redução: redução e redução através de chave (esta ultima foi bastante usada uma vez que permite indicar uma chave e realizar uma operação sobre todos os valores de *input* que possuam a mesma chave (Figura 3.4)).

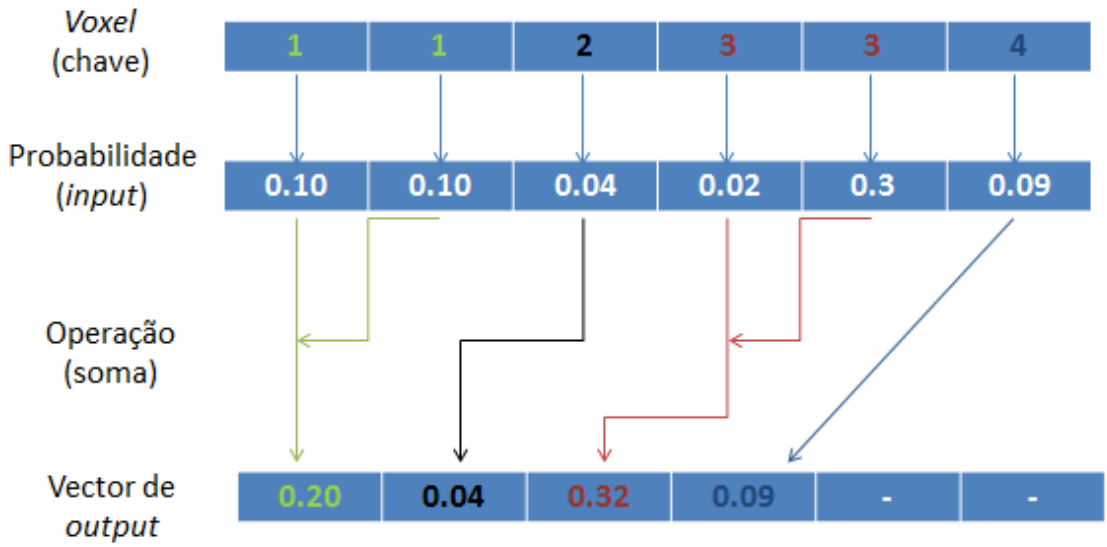

Figura 3.4 Esquema da função *reduce_by_key* da biblioteca Thrust. A figura é uma representação de um dos somatórios do algoritmo de reconstrução de imagem ($\sum_i a_{ij}$), no entanto os valores representados são aleatórios.

As funções desta biblioteca encontram-se optimizadas ao ponto de o utilizador não ter que se preocupar em definir o número de blocos e *threads* a usar na execução da função.

Actualmente estas bibliotecas (CUBLAS e Thrust, entre outras) estão inseridas no pacote de *download* do CUDA™ Toolkit.

3.2.3 Matriz de Sistema

A matriz de sistema descreve o modelo físico e geométrico dos processos de transmissão e detecção de determinado sistema. O cálculo desta matriz depende apenas da posição da fonte de raios-X e cada elemento, a_{ij}, da mesma, representa a probabilidade de os raios-X, emitidos segundo uma determinada LOR (linha definida pelo *bin* i e pela fonte de raios-X), terem sido terem sido atenuados num voxel j. O facto de a matriz de sistema ser dependente da posição da fonte de raios-X faz com que para cada projecção angular, se tenha que calcular uma nova matriz de sistema.

A matriz é calculada tendo por base um método *ray driven* [92], no qual se calculam as intersecções das LORs com os eixos (x, y, z). Com essas intersecções são obtidas as distâncias euclidianas entre intersecções – após normalização representam a contribuição relativa de um voxel na atenuação dessa LOR.

O primeiro passo no cálculo da matriz de sistema passa por definir as coordenadas relativas do detector, da fonte e da FOV. Todas as medidas necessárias estão esquematizadas na Figura 3.2 sendo que a origem é definida no prato de apoio da mama e não no detector. De forma a definir as coordenadas da fonte tem-se então as seguintes relações:

- $xfocus = \frac{Dimensão\ do\ Detector\ em\ x}{2} * xbinsize = \frac{2816}{2} * 0.085\ mm$

É a única coordenada que é sempre constante, independentemente da posição da fonte (Figura 3.5). O seu valor é igual a metade do tamanho detector na direcção x, vezes o tamanho do *bin*.

- $yfocus = Distância_{eixo\ de\ rotação\ à\ fonte} * sen(\theta) + \frac{Dimensão\ do\ Detector\ em\ y}{2} * ybinsize = 650\ mm * sen(\theta) + \frac{3584}{2} * 0.085\ mm$
- $zfocus = Distância_{eixo\ de\ rotação\ à\ fonte} * cos(\theta) + Distância_{prato\ de\ apoio\ ao\ eixo\ de\ rotação} = 650\ mm * cos(\theta) + 30\ mm$

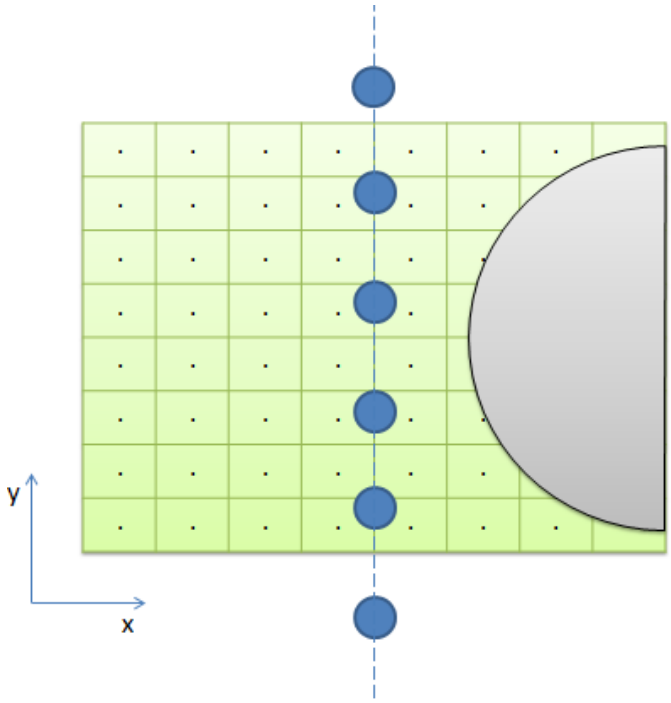

Figura 3.5 Esquema do detector, da mama e das posições da fonte.

O facto de a fonte de raios-X se manter numa posição constante ao longo do eixo dos x, permite poupar processamento no cálculo da matriz de sistema, bastando para tal, calcular a matriz de sistema de um dos lados do detector e obter o outro por simetria. Ou seja, uma LOR do lado esquerdo do detector é sempre definida por um *bin* com as coordenadas $(Detector_{Dim_x}/2 - x, y, 0)$ e uma LOR do lado direito é sempre definida por um bin com as coordenadas $(Detector_{Dim_x}/2 + x, y, 0)$. A reconstrução é feita de igual forma, independentemente do lado do detector sobre o qual se decidam efectuar os cálculos.

As LORs são então definidas através da equação da recta

$$(x, y, z) = (x_{bin}, y_{bin}, z_{bin}) + (u, v, w) * \lambda \quad \textbf{(3.1)}$$

A partir da equação da recta é possível achar as intersecções de cada LOR com os eixos (x, y, z), assumindo que a LOR intersecta os eixos sempre que um ponto da LOR for igual

a um número natural vezes o tamanho do *bin* (Na Figura 3.6 são esquematizadas as intersecções com o eixo dos x).

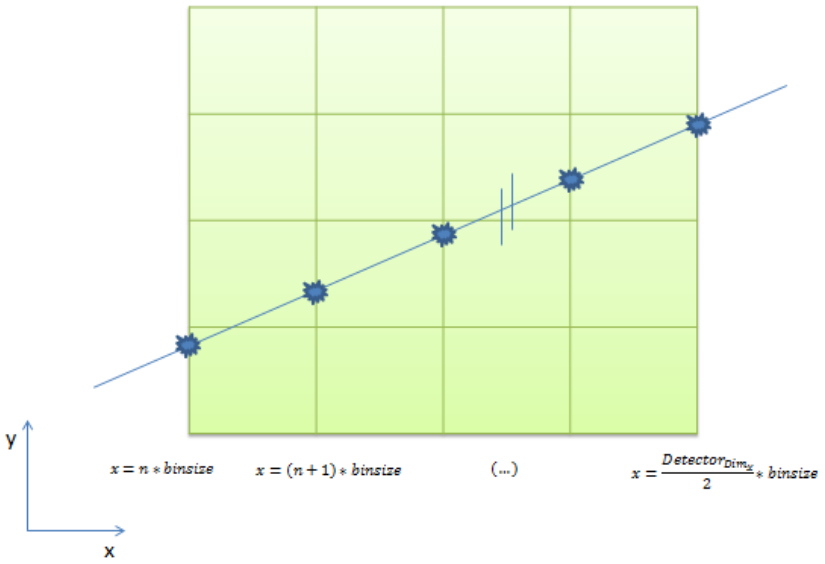

Figura 3.6 Intersecção de uma LOR com o eixo dos x

Desta forma é possível achar o parâmetro λ, que nos permite calcular as coordenadas dos dois eixos restantes. As equações seguintes demonstram o exemplo para as intersecções com o eixo dos x:

$$\lambda = \frac{x - x_{bin}}{u} \quad \textbf{(3.2)}$$

$$y = y_{bin} + v\lambda \quad \textbf{(3.3)}$$

$$z = z_{bin} + w\lambda \quad \textbf{(3.4)}$$

Após se obterem todas as intersecções de todas as LORs com os eixos, é necessário excluir os pontos que não se encontram dentro da FOV. Dado que o comprimento de uma LOR é bastante superior às dimensões da FOV, existem muitas intersecções que vão acontecer fora desta, isto acontece porque a FOV é delimitada pela placa de compressão e a fonte de raios-X está posicionada bastante acima desta, como se pode verificar na Figura 3.2. Na Figura 3.7 é esquematizada uma LOR a atravessar a FOV. Os pontos da LOR que intersectam os eixos dentro da FOV estão representados a tracejado. Esses pontos são necessários na reconstrução da imagem, todos os que se encontram fora da FOV são eliminados.

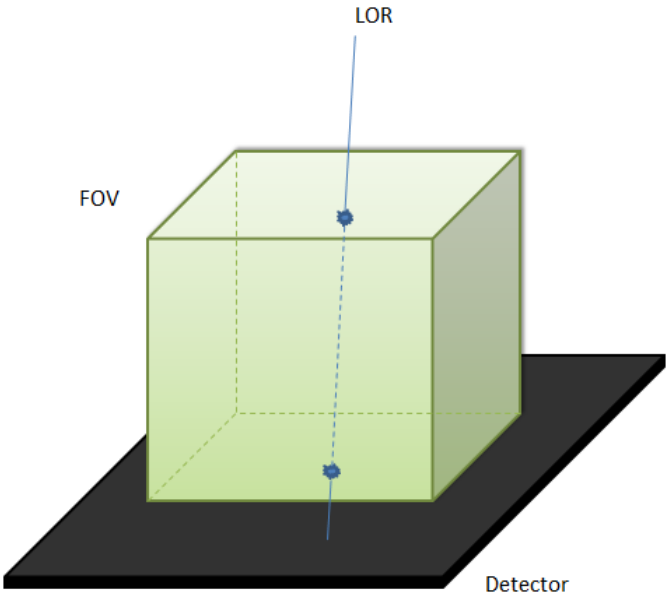

Figura 3.7 Esquema de uma LOR a atravessar a FOV. Apenas os pontos a tracejado da LOR interessam para a reconstrução, os outros são eliminados.

Por fim é necessário calcular as distâncias entre intersecções consecutivas e as coordenadas de todos os voxeis intersectados por uma LOR (essas coordenadas correspondem ao ponto no qual a LOR sai do voxel). Para tal assume-se que a LOR se inicia no *bin* e termina na fonte. Todas as LORs possuem um declive positivo no plano xz (uma vez que só se trata os dados da metade esquerda da matriz de sistema) mas o mesmo não é verdade no plano yz (o declive depende da posição da fonte).

Tendo por base estas considerações, as coordenadas são então obtidas subtraindo uma unidade de escala sempre que as coordenadas da intersecção são múltiplas da largura do *bin*. Nos casos em que as coordenadas não são múltiplos da largura do *bin*, arredondam-se por defeito. Para um declive negativo todas as coordenadas são arredondadas por defeito e no caso de serem múltiplos da largura, não sofrem alterações (Figura 3.8).

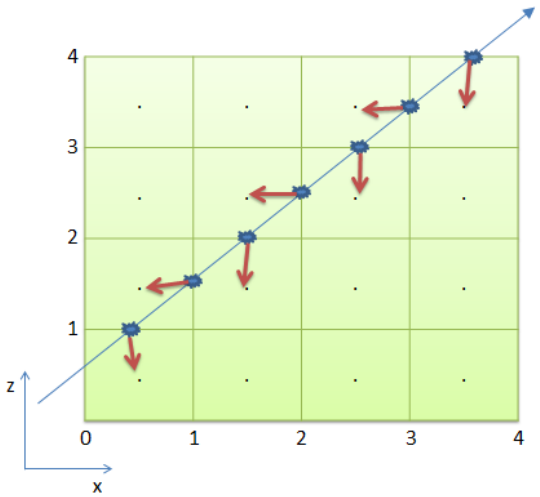

Figura 3.8 Método para calcular as coordenadas dos voxeis intersectados por uma LOR de declive positivo. Assume-se nesta figura que o comprimento da aresta do *bin* corresponde à unidade.

Para proceder à paralelização do cálculo da matriz de sistema, atribuiu-se uma *thread* a cada *bin* do detector sendo cada *thread* responsável por calcular as intersecções da LOR que esse *bin* define com a fonte de raios-X. Ao calcular as distâncias entre intersecções, cada *thread* é responsável por calcular uma distância euclidiana entre a intersecção n e $n+1$.

3.2.4 Limitações de memória

Apesar do enorme poder de processamento numérico que uma GPU possui, a sua memória é bastante limitada sendo que, actualmente, o padrão da memória das GPUs em computadores de gama média é da ordem de 1 GB. Para aplicações num campo como a imagem médica, cujos detectores possuem resoluções bastante elevadas, 1 GB está longe de ser suficiente para conter todos os dados necessários à reconstrução da imagem.

É fácil demonstrar a quantidade de informação em causa no cálculo da matriz de sistema e na implementação dos algoritmos de reconstrução. Para fazer a actualização da imagem é necessário ter na memória da GPU:

- Uma matriz $f_j^{(k-1)}$que corresponde à estimativa actual dos coeficientes de atenuação do objecto. Esta matriz tem as dimensões $Detector_{Dim_x} * Detector_{Dim_y} * NSlices * sizeof(float)$, em que $Detector_{Dim_x} * Detector_{Dim_y} * NSlices$ corresponde às dimensões da FOV e $sizeof(float)$ corresponde ao tamanho que uma variável *float* ocupa em memória (4 bytes). Tem-se portanto:

$$2816 * 3584 * 60 * 4 = 2422210560\ Bytes = 2.42\ GB.$$

- Duas matrizes com o mesmo tamanho de $f_j^{(k-1)}$, correspondentes ao factor de actualização $\sum_i \frac{a_{ij} Y_i}{\sum_i a_{ij} f_j^{(k-1)}}$ e de normalização ($\frac{1}{\sum_i a_{ij}}$) dos algoritmos de reconstrução.

Apenas estas matrizes perfazem um total de $2.42 * 3 * 10^9\ Bytes = 7.26\ GB$. Há ainda que ter em conta, a memória ocupada pela matriz de sistema e o facto de a função *sort* da biblioteca Thrust necessitar de espaço livre na memória da GPU. Esta quantidade de memória só está disponível em GPUs topo de gama que ultrapassam a barreira do milhar de euros. Uma vez que a algum ponto do algoritmo, duas dessas matrizes necessitam, obrigatoriamente, de estar na memória da GPU, foi necessário introduzir um factor de escala no programa de forma a modificar a resolução final da imagem. Assim, na reconstrução da imagem, considerou-se que cada *bin* possui a dimensão de 0,34 mm (ou seja um factor de 4x em relação aos 0,085 mm originais), passando as dimensões de reconstrução para $\frac{2816}{4} * \frac{3584}{4} * 60 * 4 = 151388160\ Bytes \approx 0.1514\ GB$. O que significa, que se se tiver as três matrizes referidas em memória, se ocupará agora, aproximadamente, $0.454\ GB$.

O cálculo da matriz de sistema foi também dividido em blocos (não de *threads*, mas de dados, ou seja blocos de *bins* do detector) de forma a limitar a memória. Estes blocos possuem as dimensões de 88x56 *bins*, o que significa que é necessário um ciclo *for* para percorrer esses mesmos blocos – 8 vezes na direcção x ($\frac{Detector_{Dim_x}}{Bloco_{Dim_x}} = \frac{704}{88} = 8$) e 16 vezes na direcção y ($\frac{Detector_{Dim_y}}{Bloco_{Dim_y}} = \frac{896}{56} = 16$). Para os cálculos de ocupação de memória da matriz de sistema foi tido em conta o máximo espaço que esta poderia ocupar, ainda antes da eliminação de pontos fora da FOV.

3.2.5 Funcionamento do programa desenvolvido

Devido à extensão do código, não se tenta nesta secção explicar a fundo o funcionamento do programa. Como tal decidiu-se representar num fluxograma o funcionamento do programa (Figura 3.9), indicando, no caso de determinada parte do programa conter código em paralelo, se foi programado em CUDA™, em Thrust ou com recurso a ambos. Convém fazer referência ao facto de haver uma etapa designada por Actualização da imagem e outra designada por Cálculo dos factores correctivos: devido ao facto de se ter tido que dividir em blocos os cálculos das intersecções das LORs com os eixos $x, y\ e\ z$ (devido às limitações de memória referidas anteriormente), a actualização da imagem é feita apenas no fim do cálculo do mapa de erros de uma determinada estimativa do algoritmo. Ou seja, uma subiteração está completa apenas após se percorrem todos os blocos de dados (88x56 *bins*). No fim dessa subiteração a imagem é então actualizada, e é necessário percorrer novamente todos os blocos de dados e usar a imagem actualizada como estimativa dos coeficientes de atenuação para a próxima subiteração. De forma a se ter em conta as contribuições de todos estes blocos de dados, criaram-se duas variáveis respeitantes a duas partes do algoritmo: actualização ($\sum_{i \in S_n} \frac{a_{ij} Y_i}{\sum_t a_{it} \lambda_t^n}$) e normalização ($\frac{1}{\sum_{i \in S_n} a_{ij}}$). A cada iteração dos blocos de dados são somadas as contribuições de cada bloco para estas variáveis. Após se percorrerem todos os blocos, multiplicam-se essas variáveis pela estimativa actual. Estas variáveis são depois reinicializadas a zero na subiteração seguinte. O cálculo destes somatórios é designado no fluxograma, como Cálculo dos factores correctivos.

Para uma visualização mais detalhada do programa, é apresentado no Anexo I, um pseudo-código que engloba as fases mais importantes de todo o código.

A linguagem de programação na qual o código de CPU foi implementado, IDL, possui muitas funções próprias da linguagem, pelo que ao programar em CUDA C foi necessário programar muitas dessas funções de raiz, que podem não atingir o nível de optimização das suas congéneres em IDL, o que pode provocar alterações ao nível de desempenho e execução do código.

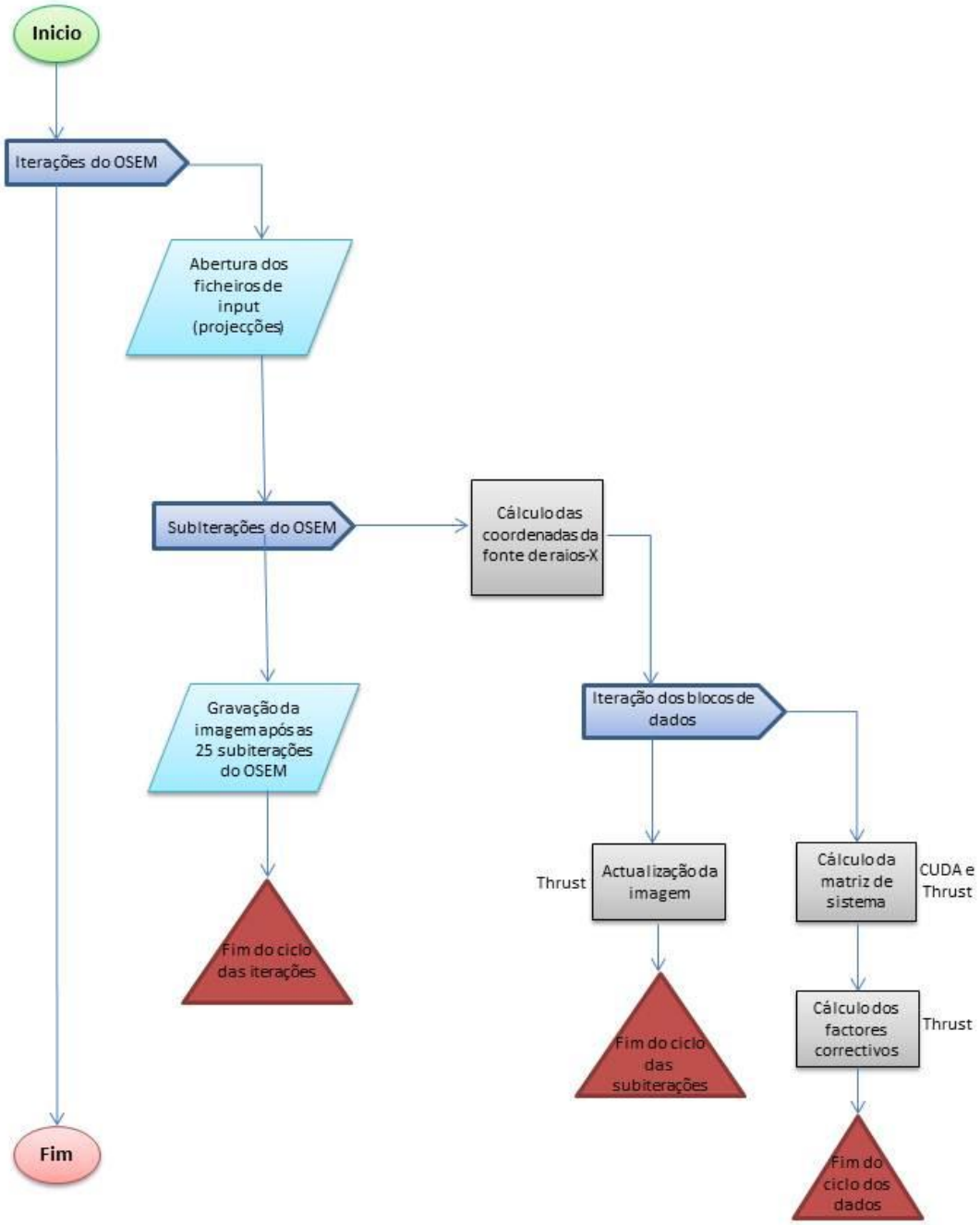

Figura 3.9 Fluxograma do funcionamento do programa

Capítulo 4

Resultados e Discussão

4.1 Resultados

Como foi referido anteriormente, neste trabalho implementaram-se os algoritmos iterativos de reconstrução de imagem tendo em conta a formulação matemática para tomografia de emissão. Esta escolha de implementação teve como base o facto de haver uma implementação prévia em CPU desses mesmos algoritmos, permitindo uma melhor comparação, quer a nível da qualidade de imagem, quer, principalmente, a nível dos tempos de reconstrução da imagem, sendo este o objectivo principal do trabalho.

Devido ao facto de a tomografia com amplitude angular limitada exibir artefactos na direcção $\boldsymbol{z}$ [93], provocado por uma desfocagem das estruturas em planos da imagem adjacentes e também devido ao facto de estes mesmos artefactos aumentarem de intensidade à medida que os planos da imagem se aproximam dos limites inferiores e superiores de $\boldsymbol{z}$, analisaram-se imagens mais próximas do eixo de rotação. Na Figura 4.1 é possível observar quatro cortes da imagem obtidos com a implementação em CPU e GPU.

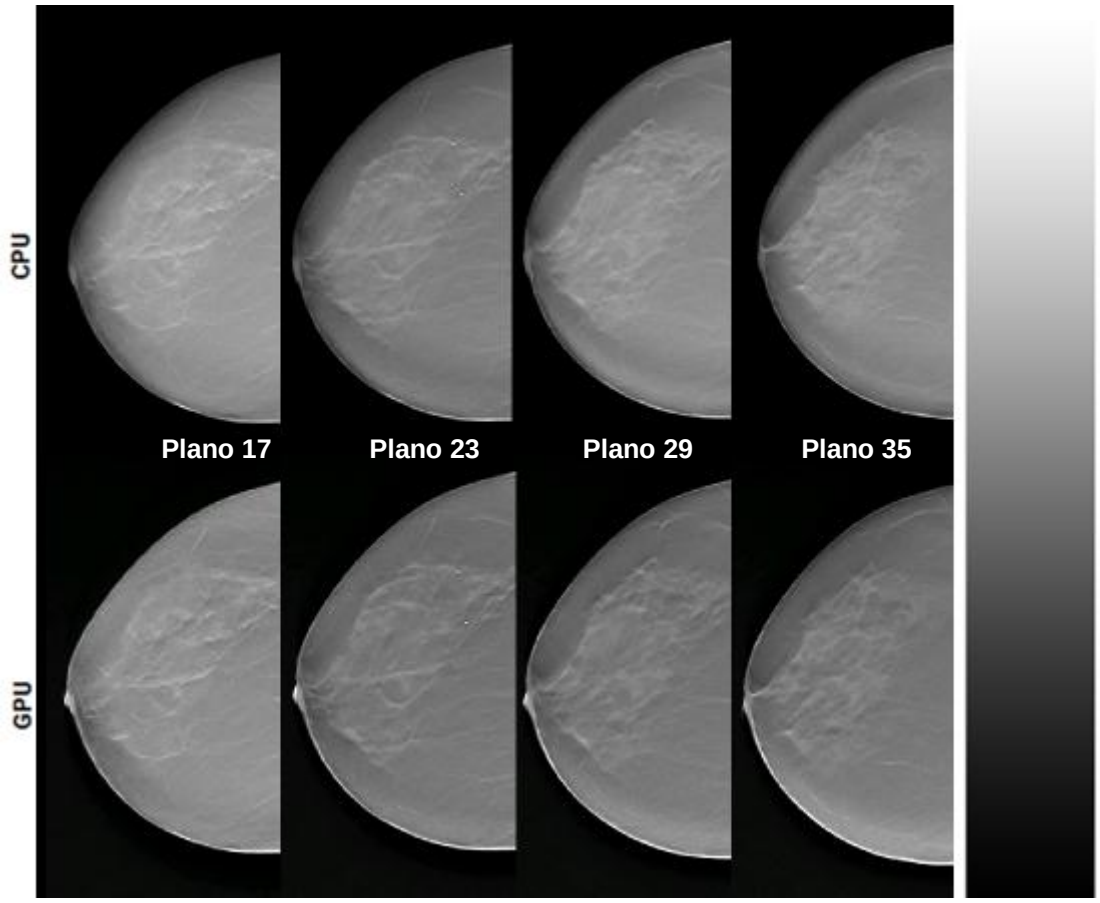

Figura 4.1 Diferentes planos (cortes) verticais da imagem obtida em CPU (em cima) e em GPU (em baixo). Imagens obtidas após 25 subiterações do algoritmo OS-EM, ou seja, após todas as projecções.

Comparando as imagens obtidas através das duas implementações, é difícil discernir diferenças significativas entre ambas as implementações, apesar de ser possível visualizar algumas diferenças nomeadamente na zona do mamilo e no contorno da mama. É de salientar, no entanto, que na implementação em IDL é usada uma máscara que permite a reconstrução da imagem apenas na zona da mama, o que permite uma redução dos artefactos (nomeadamente os que são visíveis à medida que se afasta do eixo de rotação, como foi referido anteriormente). As imagens da Figura 4.1 são gravadas em formato *.PNG*, que comprime, de maneira bastante acentuada a imagem (as imagens originais têm tamanhos da ordem dos 100MB, as imagens aqui apresentadas têm tamanhos da ordem dos 10KB). Como tal, estas imagens não são representativas do seu verdadeiro valor como ferramenta de diagnóstico médico.

Relativamente ao ML-EM, os resultados obtidos não foram satisfatórios, como pode ser observado na Figura 4.2. A imagem obtida apresenta um nível de detalhe semelhante à sua congénere de OS-EM no entanto apresenta também um artefacto que dificulta a análise da imagem. Devido ao facto de as imagens disponíveis da reconstrução em CPU, terem sido obtidas através do OS-EM de 25 subiterações, não foi possível confirmar se este artefacto advém de algum problema no código, ou se é um problema inerente ao uso do algoritmo para tomografia de emissão e não de transmissão. Este problema, levou a que a comparação de

desempenho e qualidade de imagem, fosse feita com recurso a imagens obtidas a partir do OS-EM de 25 subiterações nas duas implementações.

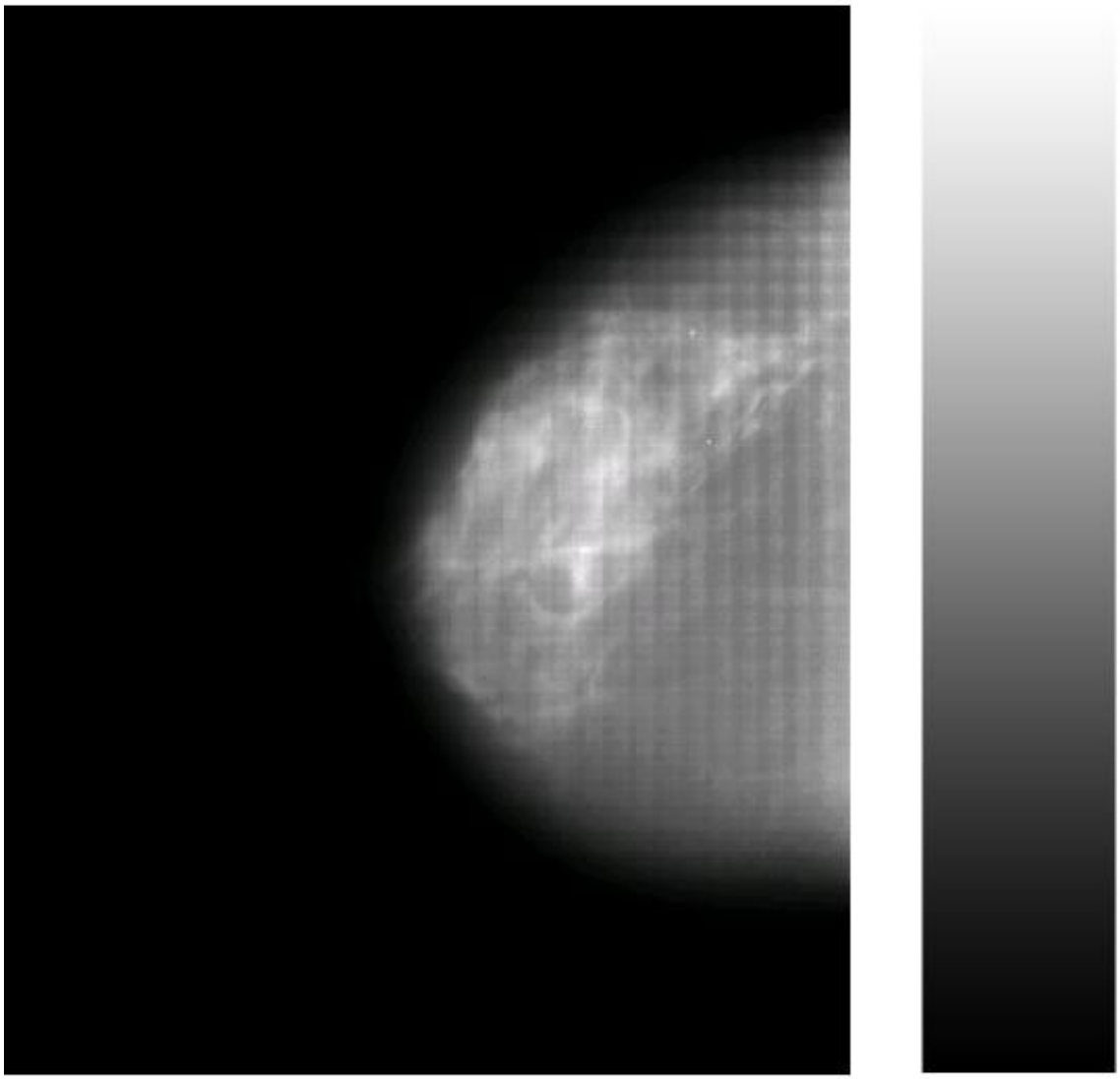

Figura 4.2 Imagem obtida através da implementação do ML-EM em GPU (plano 23). É possível observar um artefacto que dificulta a análise da imagem

4.1.1 Modelo dos processos de transmissão e detecção

Como referido anteriormente, a matriz de sistema é uma descrição geométrica dos processos físicos de detecção e emissão e como tal, uma forma de validar a sua implementação passa por verificar se a posição relativa de objectos de referência se mantém nas duas implementações. Assim para analisar a sua implementação compararam-se as distâncias relativas entre duas microcalcificações presentes no exame e a distância individual de cada uma delas à origem do referencial da imagem, conforme esquematizado na Figura 4.3. Essas distâncias são depois comparadas nas duas implementações, de forma a se confirmar a correcta implementação da matriz de sistema.

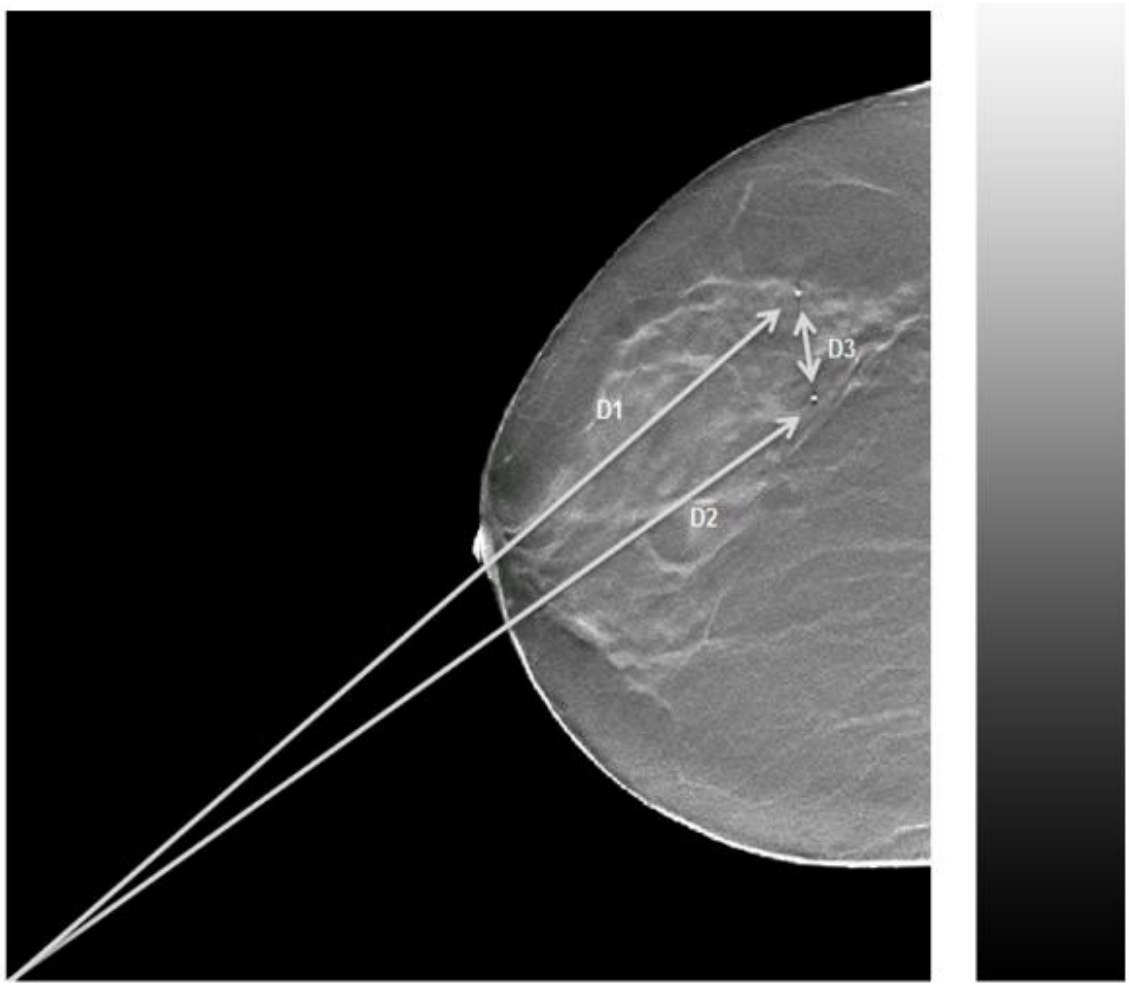

Figura 4.3 Distâncias relativas medidas para analisar a implementação da matriz de sistema. Plano 23/60 em ambas as imagens (Imagem em cima correspondente à reconstrução em GPU).

Tabela 4-1 Distâncias relativas de microcalcificações em GPU e CPU.

	Implementação em GPU	Implementação em CPU	Variação
D1	240,10 mm	239,82 mm	0,12 %
D2	225,17 mm	224,57 mm	0,27 %
D3	23,76 mm	23,72 mm	0,17 %

As distâncias medidas representam uma diferença inferior a 0,5%, pelo que se demonstra que o modelo de detecção e transmissão (i.e. matriz de sistema) foi bem implementado.

4.1.2 Análise do tempo de reconstrução da imagem

O principal objectivo deste trabalho é avaliar a possibilidade de diminuição do tempo de reconstrução das imagens de DBT com recurso à GPU. Pretende-se fazer a reconstrução de imagem com recurso a algoritmos iterativos, num tempo compatível com a prática clinica.

A comparação do tempo de reconstrução da imagem, foi feita tendo em conta uma implementação do OS-EM na linguagem IDL num CPU Intel® Xeon® E5530 2,4 GHz a correr o sistema operativo Red Hat Enterprise Linux™ 5.3 comparativamente à implementação do mesmos algoritmo em CUDA™ no GPU NVIDIA® Quadro® FX 3800 1GB (O CPU usado foi o mesmo em ambas as implementações). Os processos do computador foram mantidos ao mínimo, sendo a ocupação da CPU e GPU feita, quase exclusivamente, pela execução do programa. Em casos que tal não aconteça, o tempo de reconstrução pode aumentar.

Os tempos de reconstrução foram medidos para um algoritmo OS-EM com 25 subconjuntos, ou seja, é feita a actualização da imagem a cada projecção.

Tabela 4-2 Tempos de reconstrução da imagem em GPU e em CPU e o *speedup* obtido.

	Tempo de reconstrução (segundos)	***Speedup*** $(\frac{t_{CPU}}{t_{GPU}})$
GPU	864	6,18x
CPU	5340	

Os resultados estão de acordo com os resultados previstos, ou seja, uma redução significativa (aproximadamente 6,2 vezes) no tempo de reconstrução[11].

4.1.3 Avaliação quantitativa da qualidade de imagem

Como referido na secção dos métodos, foi usado uma variante do fantoma Gammex 156 (Figura 4.4) para a análise quantitativa da qualidade de imagem. Para essa avaliação foi usado o *software* QuasiManager, uma vez que possui funções que permitem marcar Regiões de Interesse (ROI) e obter várias medidas (como por exemplo o valor em cada voxel, a média e a variância) dessa mesma ROI.

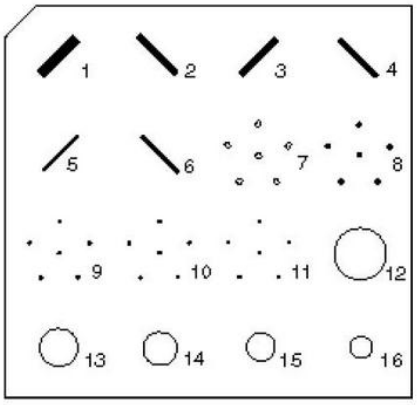

Figura 4.4 Fantoma Gammex 156

[11] O tempo de reconstrução pode oscilar consoante os processos a decorrer no computador. O valor aqui observado corresponde ao máximo speedup que se conseguiu registar

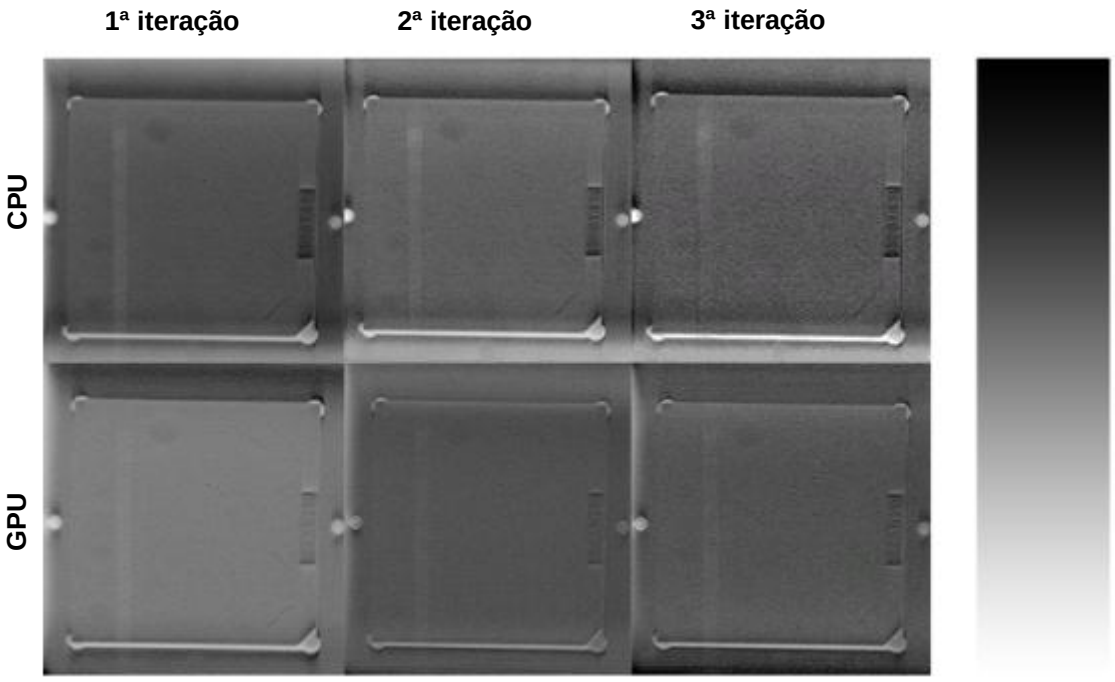

Figura 4.5 Imagem reconstruída do fantoma em CPU (cima) e GPU (baixo)

Devido ao facto de o fantoma não ser representativo de uma mama, as ROIs têm que ser marcadas em zonas que sejam distinguíveis do fundo. Assim sendo, para esta avaliação marca-se uma ROI na zona de interesse e outras ROIs no fundo, como ilustrado na Figura 4.6. A marcação de várias ROIs de fundo serve para obter uma amostra mais representativa do fundo.

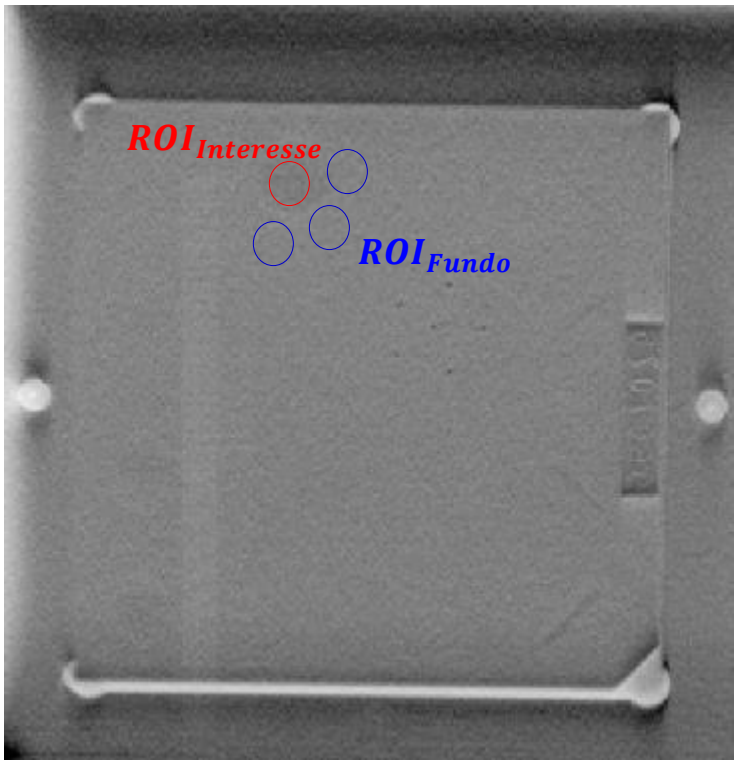

Figura 4.6 Marcação de ROIs para a análise quantitativa das imagens do fantoma

Os parâmetros quantitativos analisados (ruído, SNR, resolução espacial e contraste) foram medidos sempre em relação a uma imagem reconstruída após uma iteração completa (composta por 25 subiterações) do OS-EM. Devido à rápida convergência do algoritmo de reconstrução, esta análise foi realizada apenas para 3 iterações, uma vez que a partir da primeira iteração a imagem sofre uma degradação continua (mais visível na reconstrução em CPU).

4.1.3.1 Razão Sinal-Ruído

O cálculo da SNR é feito usando a equação 4.1, onde $Média_{Região}$ é a média do valor dos voxeis na ROI de interesse, $DesvioPadrão_{Fundo}$ é o desvio padrão da variância média dos voxeis das ROIs de fundo e $Média_{Fundo}$ é a média do valor dos voxeis nessas mesmas ROIs.

$$SNR = \frac{Média_{Região} - Média_{Fundo}}{DesvioPadrão_{Fundo}} \quad \textbf{(4.1)}$$

A SNR é um parâmetro que ajuda a ter uma percepção de como o sinal se impõe sobre um determinado nível de ruído. Para um objecto ser identificável a sua SNR tem que ser superior a 5 (limite de detectabilidade). Este limite de detectabilidade é o valor convencionalmente usado em mamografia de raios-X [94]. Um valor de SNR inferior a 5 não significa que o objecto não seja identificado, mas sim que a probabilidade de não o ser deixa de ser negligenciável [95].

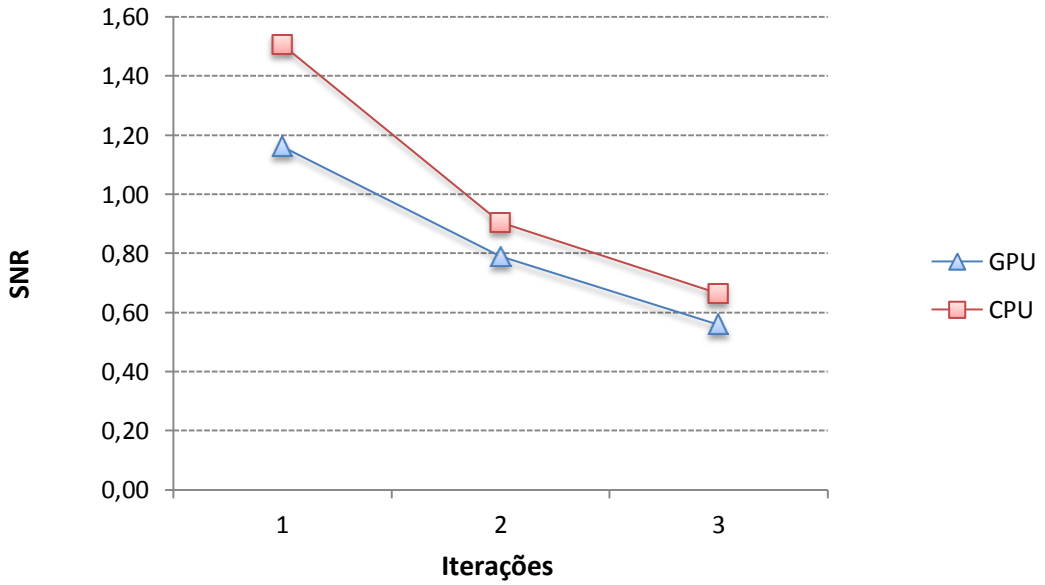

Figura 4.7 Gráfico da razão sinal-ruído em função do número de iterações

Os valores obtidos são idênticos para CPU e GPU, no entanto, a SNR é ligeiramente superior na CPU. A SNR diminui conforme o número de iterações o que representa uma

degradação contínua da imagem com o aumento do número de iterações. Os valores obtidos são inferiores ao limite de detectabilidade, no entanto, convém referir que a estrutura de interesse delineada pela ROI não é perfeitamente discernível. Dado que a SNR envolve tanto a diferença das intensidades com que as estruturas são visualizadas, tal como o ruído, analisaram-se separadamente as componentes de contraste e ruído.

4.1.3.2 Ruído

O ruído aqui estudado refere-se ao ruído relativo que é o ruído perceptível por um observador humano. O cálculo do ruído é feito a partir do Coeficiente de Variação (CV) e é dado pela equação 4.2

$$CV = \frac{DesvioPadrão_{Fundo}}{Média_{Fundo}} \quad \textbf{(4.2)}$$

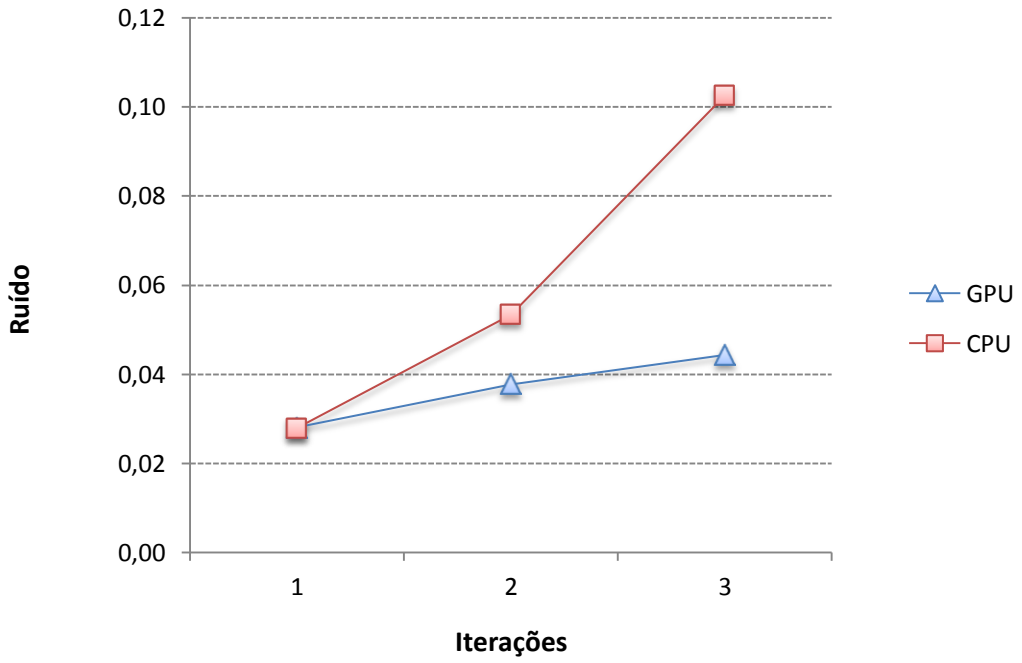

Figura 4.8 Gráfico do ruído em função do número de iterações

O ruído aumentou com o número de iterações o que se reflecte directamente na visualização das imagens, sendo o aumento de ruído mais notório na reconstrução efectuada na CPU (o que é observável por análise das imagens da Figura 4.5).

4.1.3.3 Contraste

O contraste pode ser definido como a diferença de intensidade entre duas regiões adjacentes [95]. O contraste é calculado tendo por base a equação 4.3

$$Contraste = \frac{Média_{Região} - Média_{Fundo}}{Média_{Fundo}} \quad \textbf{(4.3)}$$

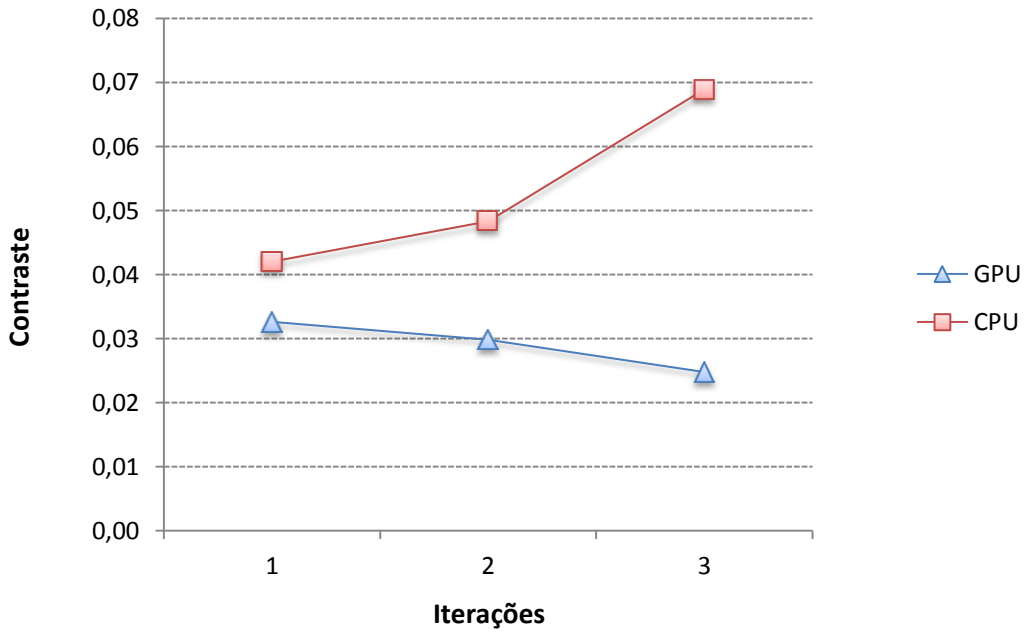

Figura 4.9 Gráfico do contraste em função do número de iterações

Os valores obtidos para o contraste foram algo díspares quando comparando a reconstrução em CPU (aumenta significativamente com o número de iterações) com a reconstrução em GPU (diminui ligeiramente com o número de iterações). No entanto, comparando as imagens da Figura 4.5, há uma melhor percepção de contraste na imagem de CPU, o que vai de encontro aos valores obtidos, apesar de na primeira iteração, a diferença ser reduzida. O superior contraste da imagem em CPU, valida o facto de a SNR ser superior na imagem obtida em CPU, apesar de o ruído ser mais significativo do que em GPU.

4.1.3.4 Resolução Espacial

A resolução espacial está directamente relacionada com o nível de detalhe da imagem e a sua definição clássica baseia-se na medida da distância mínima a que dois objectos conseguem ser vistos como distintos. Para o cálculo da resolução espacial, mediu-se a largura a meia altura (FWHM) das curvas gaussianas ajustadas aos perfis medidos numa estrutura pontual como ilustrado na Figura 4.10.

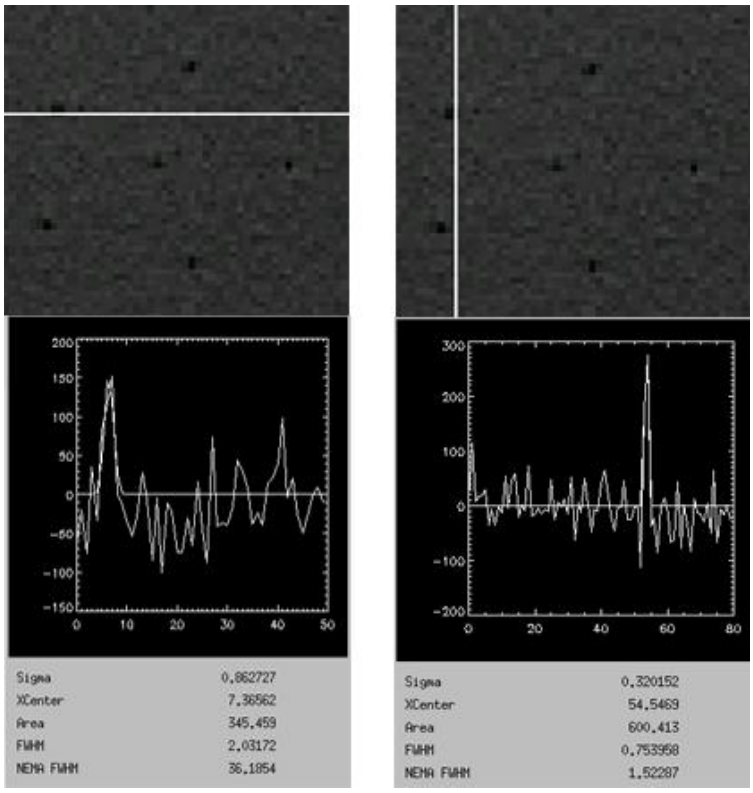

Figura 4.10 Marcação de um perfil sobre uma estrutura pontual na direcção x (esquerda) e na direcção y (direita).

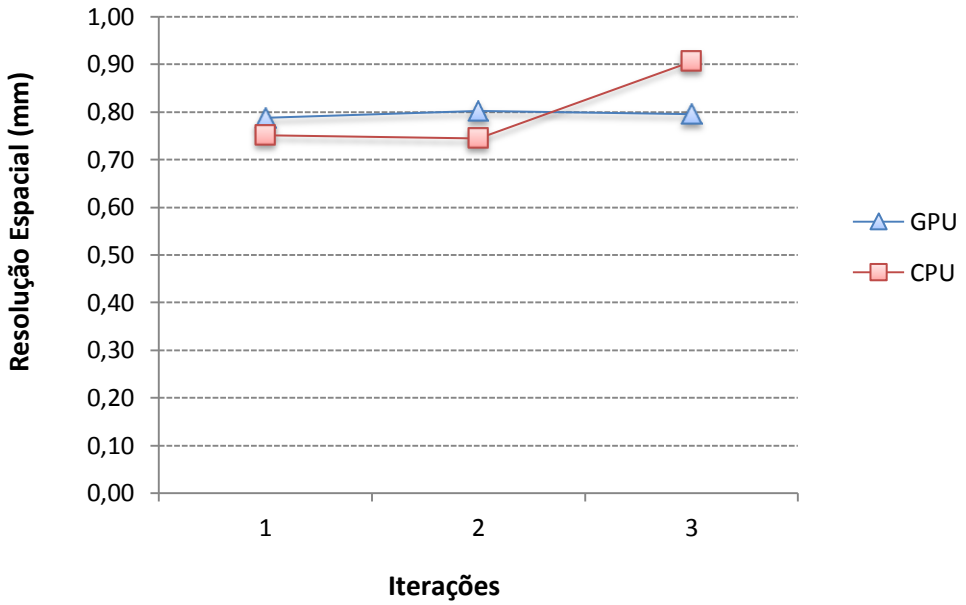

Figura 4.11 Gráfico da resolução espacial em x em função do número de iterações

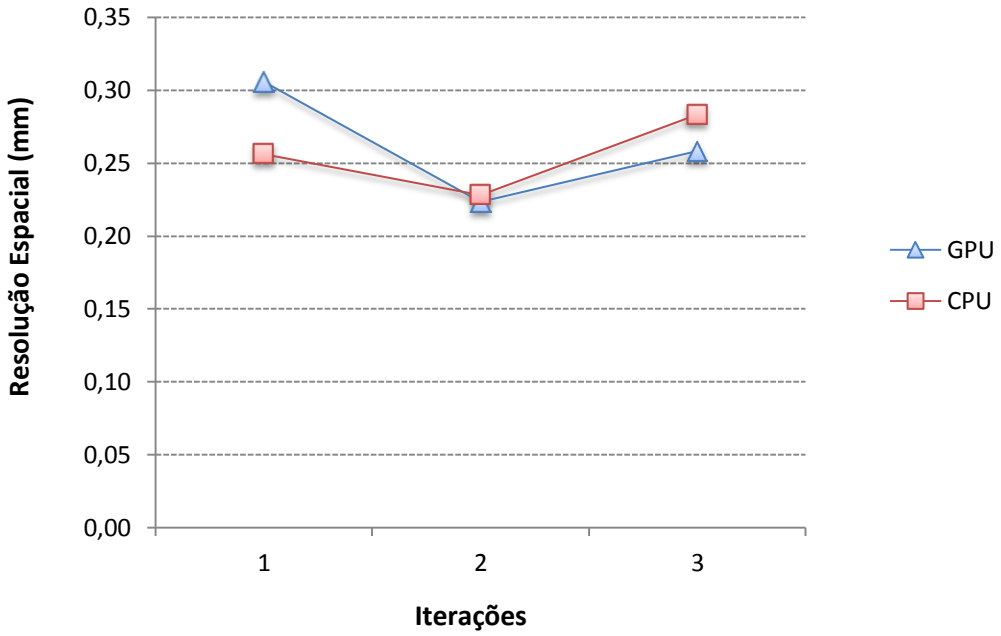

Figura 4.12 Gráfico da resolução espacial em y em função do número de iterações

A resolução espacial medida para os dois eixos (Figura 4.11 e Figura 4.12), não sofreu uma variação muito significativa com o número de iterações e os valores encontrados, são semelhantes para as duas implementações.

4.2 Discussão

Os resultados obtidos são, no geral, positivos. Relativamente ao desempenho do algoritmo, os resultados foram muito relevantes do ponto de vista da potencial aplicação clínica: uma redução do tempo de reconstrução de aproximadamente 6,2 vezes (cerca de 15 minutos para a reconstrução em GPU e cerca de 1h30m para a reconstrução em CPU). A comparação dos resultados obtidos em termos de desempenho, não pode ser feita de forma directa relativamente aos vários estudos da literatura referidos anteriormente, devido à diversidade de algoritmos usados para diferentes técnicas de imagem médica, dos volumes de reconstrução e dos diferentes GPUs que se utilizaram em cada trabalho, sendo óbvia, por exemplo, a vantagem, que um dispositivo multi-GPU tem sobre um dispositivo de GPU única, ou que um volume de reconstrução de 100x100x100 tem sobre um volume de 500x500x500.

No entanto convém referir que o melhoramento do desempenho poderia atingir números mais expressivos, sendo limitada por factores de memória que foram referidos na secção correspondente: o facto de se ter que dividir os dados em blocos e percorrer os mesmos através de um ciclo *for*, aumenta a proporção de código sequencial. Através da análise feita, nas considerações teóricas, à lei de Amdahl (equação 2.13), é fácil verificar que uma parcela reduzida de código sequencial, impõe limites significativos nos *speedups* que é possível obter, senão veja-se:

- Considerando uma porção de código paralelo que perfaz 99% do código total, o *speedup* teórico que se pode obter, é dado substituindo P por 0.99 na equação 2.14:

$$Speedup = \frac{1}{1 - 0.99} = 100x$$

- Se a porção de código paralelo diminuir para 95% o *speedup* diminui consideravelmente:

$$Speedup = \frac{1}{1 - 0.95} = 20x$$

Ou seja, uma redução de 4% da porção de código paralelo implica uma diminuição de 80% do *speedup* em relação ao valor inicial.

É então possível concluir que usando uma GPU com mais memória é possível percorrer os dados em menos blocos de dados (ou mesmo processá-los todos aos mesmo tempo, no caso de GPUs topos de gama), o que provoca uma diminuição do código sequencial e consequente aumento de *speedup*.

Não foi feita a mesma análise das imagens obtidas através de ML-EM e de OS-EM, uma vez que as imagens obtidas por ML-EM apresentam artefactos que tornam a sua análise bastante difícil. Acrescenta ainda o facto de a implementação em CPU consistir apenas na

implementação do OS-EM com 25 subiterações, pelo que não foi possível fazer uma comparação com o ML-EM.

Analisando a distância relativa entre as duas microcalcificações e entre estas e a origem, foi possível confirmar a correcta implementação da matriz de sistema, sendo o erro associado a essas distâncias, inferior a 0,5%.

Na comparação da análise quantitativa das imagens, verificou-se que as duas implementações apresentaram resultados semelhantes, sendo que era expectável a existência de ligeiras disparidades uma vez que a implementação do código não se baseou numa tradução directa da implementação existente em CPU. Porém, verifica-se que os resultados obtidos com a CPU são ligeiramente superiores, sendo discernível por inspecção visual, que a imagem em CPU é ligeiramente melhor (no que à primeira iteração diz respeito uma vez que, apesar de apresentar uma melhor SNR ao longo das restantes iterações, a imagem em CPU sofre um agravamento muito grande do ruído o que torna a análise das imagens mais difícil, principalmente na terceira iteração). O aumento mais ligeiro do ruído, e a diminuição também, não muito acentuada do contraste, na reconstrução em GPU, demonstra que a imagem em GPU se degrada de forma menos acentuada do que em CPU, o que é observável na Figura 4.5.

Analisando os resultados da resolução espacial, observa-se que não há grande comprometimento da mesma, com o número de iterações ou com o tipo de implementação, sendo os resultados semelhantes. No entanto, os valores encontrados para a resolução espacial em y, são ligeiramente inferiores à largura do voxel (0,34 mm). Estes valores, que se encontram abaixo do esperado, devem-se provavelmente a aproximações no cálculo da FWHM da curva gaussiana, pelo *software* QuasiManager.

Usando os resultados do trabalho de Pratx *et al.* [71] como comparação[12], as diferenças nesta análise deviam ser menores como pode ser observado na Figura 4.13, onde é feita uma comparação do contraste consoante o ruído da imagem, entre CPU e GPU, e na qual os resultados das duas implementações são indistinguíveis.

[12] O algoritmo iterativo usado, é uma variante do OS-EM (*List-mode* OS-EM) e a reconstrução da imagem é feita para PET

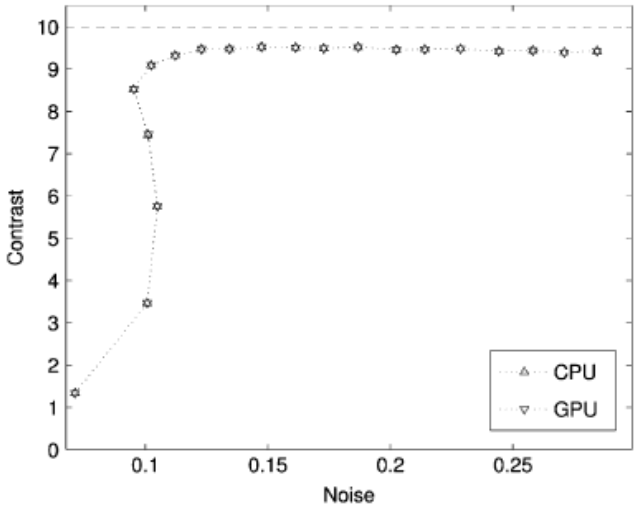

Figura 4.13 Comparação de uma implementação de um algoritmo iterativo de reconstrução de imagem para PET, em termos de contraste *vs* ruído, consoante o número de subiterações [71].

Algumas das diferenças observadas na análise quantitativa poderão advir do facto da reconstrução em IDL usar uma máscara que permite a reconstrução apenas da zona do fantoma e não do detector todo. Como foi referido anteriormente, o facto de o código desenvolvido em GPU, não ser uma tradução directa do código sequencial, mas ser feito de raiz, pode provocar algumas diferenças na execução do mesmo, o que também pode ser uma das causas para as diferenças verificadas (por exemplo, uma diferença num único ponto pode significar uma alteração na escala de cinzentos, o que tem repercussões directas na comparação das imagens). Estas diferenças estão também patentes na comparação dos perfis horizontais de um mesmo ponto em GPU e CPU (Figura 4.14). O valor dos coeficientes de atenuação em CPU são muito superiores aos de GPU, no entanto, tal acontece, devido a se adicionar, ao algoritmo, uma constante de escala na implementação em CPU, o que não tem implicações na visualização da imagem. É visível que o perfil da imagem obtida em CPU é melhor delineado, o que se deve, muito provavelmente, ao uso da já referida máscara de reconstrução. Analisando os perfis, também se consegue discernir uma ligeira diferença na forma dos picos. Estas duas diferenças verificadas através da comparação dos perfis são representativas das diferenças verificadas por inspecção visual.

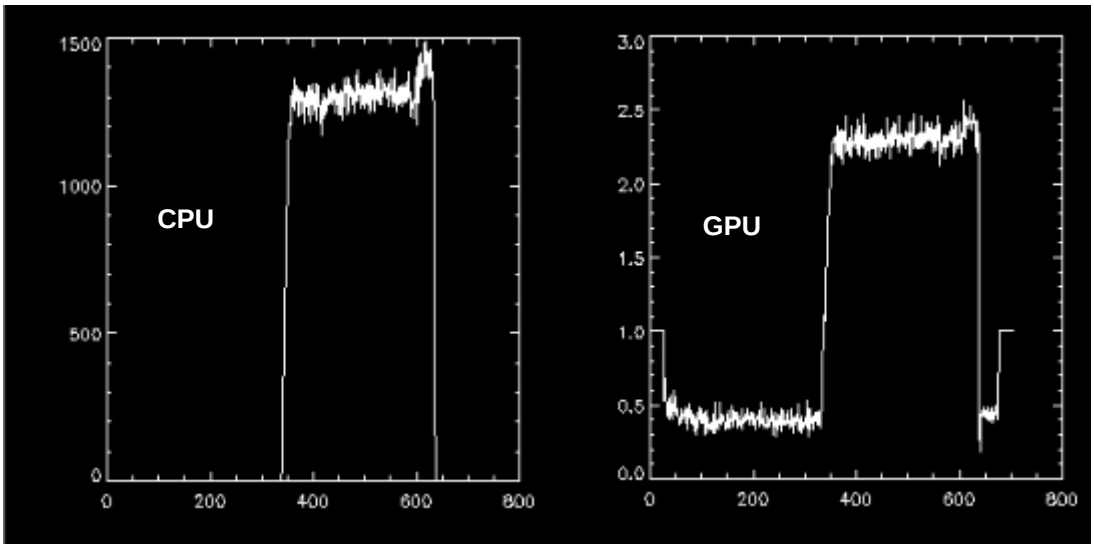

Figura 4.14 Comparação dos perfis horizontais de um mesmo ponto.

Como também já foi referido, a linguagem IDL contem na sua sintaxe funções optimizadas e bastante abrangentes. No desenvolvimento de código em CUDA C é necessário programar todas as funções que são necessárias à execução do código, como tal, a performance dessas mesmas funções e a sua eficácia, estão dependentes da experiência do programador e não da linguagem.

Analisando os resultados das imagens obtidas e da análise quantitativa das imagens é possível verificar que ambos os algoritmos (CPU e GPU) convergem bastante rapidamente, obtendo-se a melhor imagem logo após a primeira iteração. Este facto é vantajoso, uma vez que o tempo de reconstrução fica reduzido ao tempo que demora apenas uma iteração, o que no caso da GPU, se traduz em 15 minutos para reconstrução de uma imagem.

Capítulo 5

Conclusões e Trabalho Futuro

O objectivo principal do trabalho, como foi referido, foi acelerar o processo de reconstrução de imagem de DBT, usando uma implementação em GPU (com recurso à arquitectura CUDA™), de algoritmos iterativos de reconstrução de imagem. Estes algoritmos foram, até recentemente, preteridos em prática clínica em favor dos algoritmos analíticos devido à sua simplicidade e rapidez. No entanto, os algoritmos iterativos apresentam vantagens que justificam a aceleração das respectivas implementações. Com o recurso à programação em GPU, é de esperar sempre uma diminuição, mais ou menos significativa (dependendo dos factores referidos na discussão dos resultados), do tempo de reconstrução das imagens de DBT, sem grande variação da qualidade de imagem relativamente à sua congénere de CPU [60, 65, 68, 71, 80].

Efectivamente, comparando os resultados de desempenho obtidos em GPU com os resultados obtidos em CPU, conclui-se que se conseguiu uma diminuição significativa do tempo de reconstrução (aproximadamente 6,2 vezes). De salientar, que esta diminuição poderia ser muito mais significativa: usando uma GPU com 1 GB de RAM e tendo em conta as medidas da FOV (704 x 896 x 60), torna-se bastante complicado paralelizar todos os dados que se necessitam para a reconstrução de imagem. Estas limitações de memória obrigaram a que se tivesse que percorrer através de ciclos *for*, os dados a ser paralelizáveis, o que foi demonstrado, através da lei de Amdahl, ter efeitos bastante significativos no *speedup* de um código (ainda que usando reduzidas fracções de código sequencial). A mudança de resolução da imagem final de 2416 x 3584 x 60 para 704 x 896 x 60 permitiu ultrapassar alguns problemas de memória (que foram essenciais para que se pudesse ter na memória da GPU

todas as variáveis necessárias à correcta reconstrução da imagem), no entanto, não chegou a ser o suficiente para conseguir paralelizar todos os dados. É então importante perceber um dos paradigmas da programação em paralelo: apesar de permitir um elevado número de FLOPS, a memória limitada que uma GPU possui, leva a que o programador tenha que percorrer os dados, a serem trabalhados, em blocos. Esta metodologia leva a um aumento das transferências de memória de CPU para GPU e vice-versa. Estas transferências implicam acessos de memória com elevada latência pelo que é recomendado minimizar, tanto quanto possível, as mesmas.

Existem ainda muitas outras considerações de desempenho da GPU que não foram discutidas, como optimização da calendarização de *warps* e consequente divergência de *threads* e acessos coalescidos à memória. Estes detalhes não foram todos discutidos, uma vez que a sua explicação e posterior análise poderia por si só, ser motivo de outro trabalho. Convém referir, no entanto, que nos programas em CUDA™ nem sempre é fácil optimizar esses parâmetros, exigindo competências de informática muito superiores às expectáveis dum engenheiro biomédico. Por outro lado, a biblioteca Thrust, trata de optimizar esses parâmetros, deixando apenas para o utilizador, funções simples e intuitivas de usar.

É necessário salientar ainda uma das maiores desvantagens de trabalhar com programação em paralelo: o *debug* do código em GPU. Fazer *debug* em GPU é uma tarefa bastante complicada, havendo dois programas (Paralel NSight para Windows™ e CUDA-GDB para Linux™) pouco eficazes e muito pouco *user-friendly*.

No referente à qualidade de imagem, a análise que foi feita, demonstrou que a CPU apresenta uma ligeira vantagem sendo as diferenças pouco significativas. Todos os pormenores importantes quer na reconstrução do fantoma, quer na reconstrução da mama, são discerníveis em ambas as implementações. No entanto, e segundo os resultados de alguns estudos de reconstrução de imagem em GPU [65, 71, 80], essa diferença na reconstrução, deveria ser menos significativa, do que a obtida neste trabalho.

Como trabalho futuro seria interessante aprofundar alguns aspectos. Em primeiro lugar, estudar o desempenho do código numa GPU com maior capacidade de memória (sendo necessário alterações no código para aumentar o tamanho dos blocos de dados)[13]. Com maior capacidade de memória seria possível também reconstruir a imagem com a sua resolução original. Porém, como foi analisado na secção referente às limitações de memória, a GPU a utilizar teria que possuir mais do que 7 GB de RAM, o que implicaria um investimento monetário significativo. Em segundo lugar, proceder a uma tentativa de optimização das funções programadas em CUDA™. Tem que se ter em conta, que este trabalho foi a primeira implementação de um programa em GPU e que muito do tempo despendido no trabalho, teve como objectivo a aprendizagem de um paradigma de programação completamente diferente da programação sequencial que é leccionada durante o curso. Este facto, aliando ainda a uma complexidade mais elevada da programação em paralelo, poderá influenciar certas porções de código que eventualmente poderiam ser mais optimizadas. Por último seria interessante

[13] O código foi desenhado de forma a que baste alterar as variáveis de entrada, para promover as referidas alterações no código, não sendo necessária uma reestruturação do mesmo

implementar em GPU o algoritmo desenvolvido para tomografia de emissão por Lange e Carsson [74] e comparar com o algoritmo implementado neste trabalho. O facto da imagem obtida através do ML-EM não apresentar os resultados esperados, pode dever-se à utilização do algoritmo para tomografia de emissão. No entanto, não havendo uma implementação correspondente em CPU, não foi possível verificar se os artefactos presentes em GPU, são também visíveis em CPU. Caso se verifique que o ML-EM em CPU, produz uma imagem semelhante ao OS-EM, os artefactos da imagem em GPU, podem ter tido origem num problema na implementação do código.

Os resultados obtidos estão de acordo com o esperado e mostraram que a GPU é um dispositivo de processamento numérico bastante eficaz, podendo tornar-se no padrão da programação em problemas que envolvam um conjunto de dados elevado e um processamento intensivo dos mesmos e, ainda, em situações em que o problema possa ser paralelizável[14]. Neste trabalho estas condições estavam presentes, providenciando o *background* ideal para poder programar em paralelo. A aliar ao seu poder de processamento excelente, acresce ainda a excelente razão FLOP/Custo de uma GPU com a capacidade de programação em CUDA™, o seu reduzido consumo energético e tamanho, quando comparado com *clusters* (aglomerados*)* de CPUs normalmente usados por empresas para proceder a computações intensivas como simulações meteorológicas, sismológicas, e de biologia molecular entre outros ramos profissionais.

A redução do tempo de reconstrução de imagem verificado pode vir a representar um ponto de viragem para o aumento de popularidade dos algoritmos iterativos de reconstrução de imagem em prática clínica.

[14] No caso de a GPU ter que processar uma reduzida quantidade de dados, a sua melhoria de performance em relação à CPU é quase nula.

Bibliografia

[1] Li, C., et al., *In vivo Breast Sound-Speed Imaging with Ultrasound Tomography.* Ultrasound in Medicine & Biology, 2009. **35**(10): p. 1615-1628.

[2] Ferlay, J., et al. *GLOBOCAN 2008 v1.2, Cancer Incidence and Mortality Worldwide: IARC CancerBase No. 10.* 2010 [acedido em 30 de Julho de 2010]; Dísponivel em: http://globocan.iarc.fr.

[3] Henson, D.E. and Ries, L.A., *Progress in Early Breast Cancer Detection.* Cancer 1990. **65**: p. 2155-2158.

[4] Tabar, L., et al., *Mammography service screening and mortality in breast cancer patients: 20-year follow-up before and after introduction of screening.* Lancet, 2003. **361**: p. 1405–1410.

[5] Kuhl, C.K., Kuhnb, W., and Schilda, H., *Management of women at high risk for breast cancer: New imaging beyond mammography.* The Breast, 2005. **14**(6): p. 480-486.

[6] Matela, N., *2D Iterative Image Reconstruction for a Dual Planar Detector for Positron Emission Mammography*. 2008, Universidade de Lisboa.

[7] Administration, F.a.D. *Mammography Quality Standards Act and Program - Digital Accreditation*. 2011 [acedido em 31 de Julho de 2011]; Dísponivel em: http://www.fda.gov/Radiation-EmittingProducts/MammographyQualityStandardsActandProgram/FacilityCertificationandInspection/ucm114148.htm.

[8] Hologic, I. *Hologic Receives FDA Approval for First 3-D Digital Mammography (Breast Tomosynthesis) System*. 2011 [acedido em 31 de Julho de 2011]; Dísponivel em: http://www.hologic.com/en/news-releases/173-id.234881803.html.

[9] Verdière, G.C.d., *Introduction to GPGPU, a hardware and software background.* C. R. Mecanique, 2011. **339**: p. 78–89.

[10] Turner, J.E., *Atoms, Radiation, and Radiation Protection*. Terceira ed. 2007, Oak Ridge: Wiley-VCH.

[11] Hoheisel, M., *Review of medical imaging with emphasis on X-ray detectors.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2006. **563**(1): p. 215-224.

[12] Hessenbruch, A., *A brief history of x-rays.* Endeavour, 2002. **26**(4): p. 137-141.

[13] Hendeem, W.R. and Ritenour, E.R., *Medical Imaging Physics*. Quarta ed. 2002: Wiley-Liss.

[14] Hady, M.A., et al., *Encyclopedia of Medical Devices and Instrumentation*, Wiley-Interscience, Editor. 2006, John Wiley & Sons, Inc.,.

[15] Podgorsak, E.B., *Radiation Physics for Medical Physicists*. Primeira ed. 2006: Springer.

[16] Enderle, J.D., Blanchard, S.M., and Bronzino, J.D., *Introduction to Biomedical Engineering*. Segunda ed. 2005: Elsevier Academic Press.

[17] Eisberg, R. and Resnick, R., *Física Quântica - Átomos, Moléculas, Sólidos, Núcleos e Partículas*. Vigésima Sexta ed. 1974: Elsevier.

[18] Powsner, R.A. and Powsner, E.R., *Essential Nuclear Medicine Physics*. 2006: Blackwell Publishing.

[19] Schulz-Wendtland, R., et al., *Digital mammography: An update.* European Journal of Radiology, 2009(72): p. 258–265.

[20] Lawaczeck, R., Rein, V., and Deeg, W., *Dedicated mammography: Imaging with monochromatic X-rays and a clinical mammography unit.* Nuclear Instruments and Methods in Physics Research Section A, 2006. **548**: p. 147–154.

[21] Yaffe, M.J., *Digital mammography - detector considerations and new applications.* Nuclear Instruments and Methods in Physics Research A, 2001. **471**: p. 6–11.

[22] Simonetti, G., et al., *What's new in mammography.* European Journal of Radiology, 1998. **27**(2): p. 234-241.

[23] Siemens. *MAMMOMAT 3000 Nova*. 2011 [acedido em 2 de Agosto de 2011].

[24] H.R. Laine, T.T., P. Mikkola, T. Holmström, *Assessment of mammography and ultrasound examination in the diagnosis of breast cancer.* European Journal of Ultrasound, 1996(3): p. 9-14.

[25] Smith, A., *Fundamentals of Breast Tomosynthesis: Improving the Performance of Mammography*. 2008, Hologic.

[26] Park, J.M., et al., *Breast Tomosynthesis: Present Considerations and Future Applications.* RadioGraphics, 2007(27): p. 231-240.
[27] Sickles, E.A., Wolverton, D.E., and Dee, K.E., *Performance Parameters for Screening and Diagnostic Mammography: Specialist and General Radiologists.* Radiology, 2002(224): p. 861–869.
[28] Lindfors, K.K., O'Connor, J., and Parker, R.A., *False-Positive Screening Mammograms: Effect of Immediate versus Later Work-up on Patient Stress.* Radiology, 2001. **218**: p. 247-253.
[29] Elmore, G., et al., *Ten-year risk of false positive screening mammograms and clinical breast examinations.* New England Journal of Medicine, 1998. **338**(16): p. 1089-1096.
[30] Darambara, D.G., Taibi, A., and Speller, R.D., *Image-quality performance of an a-Si : H-based X-ray imaging system for digital mammography.* Nuclear Instruments and Methods in Physics Research A 2002. **477**: p. 521–526.
[31] James, J.J., *The current status of digital mammography.* Clinical Radiology, 2004. **59**: p. 1-10.
[32] Fischer, U., Hermann, K., and Baum, F., *Digital mammography: current state and future aspects.* European Radiology, 2006. **16**(1): p. 38-44.
[33] Belev, G. and Kasap, S.O., *Amorphous selenium as an X-ray photoconductor.* Journal of Non-Crystalline Solids, 2004. **345-346**: p. 484-488.
[34] Birdwell, R.L., Ikeda, D.M., and Shoughnessy, K.F.O., *Mammographic characteristics of 115 missed cancers later detected with screening Mammography and potential utility of computer-aided detection.* Radiology, 2001. **219**: p. 192-202.
[35] Freer, T.W. and Ullssey, M.J., *Screening mammography with computer-aided detection: prospektive study of 12.860 patients in a community breast center.* Radiology, 2001. **220**: p. 781-786.
[36] Funovics, M., Schamp, S., and Helbich, T.H., *Evaluation eines computerassistierten Diagnosesystemsinder Erkennung des Mammakarzinoms.* Fortschr Röntgenstr, 2001. **173**: p. 218-223.
[37] Malich, A., et al., *Tumour detection rate of a new commercially available computer-aided detection system.* European Journal of Radiology, 2001. **11**: p. 2454-2459.
[38] Gennaro, G., et al., *Digital breast tomosynthesis versus digital mammography: a clinical performance study.* European Radiology, 2010. **20**(7): p. 1545-1553.
[39] Felix Diekmann, U.B., *Breast Tomosynthesis.* Seminars in Ultrasound CT and MRI, 2011. **32**: p. 281-287.
[40] Bansal, G.J. and Thomas, K.G., *Imaging techniques in breast cancer.* Surgery 2010. **28**(3): p. 117-124.
[41] Zhou, W.H., et al., *Multi-beam X-ray source breast tomosynthesis reconstruction with different algorithms.* Medical Imaging 2010: Physics of Medical Imaging, 2010. **7622**.
[42] Gur, D., et al., *Digital Breast Tomosynthesis: Observer Performance Study.* American Journal of Roentgenology, 2009. **193**(2): p. 586-591.
[43] Yang, W.T., *Emerging Techniques and Molecular Imaging in Breast Cancer.* Seminars in Ultrasound CT and MRI, 2011. **32**: p. 288-299.
[44] Vecchio, S., et al., *A novel approach to digital breast tomosynthesis for simultaneous acquisition of 2D and 3D images.* European Journal of Radiology, 2011. **21**: p. 1207–1213.
[45] Teertstra, H.J., et al., *Breast tomosynthesis in clinical practice: initial results.* European Radiology, 2010. **20**(1): p. 16-24.
[46] Niklason, L.T., Christian, B.T., and Niklason, L.E., *Digital tomosynthesis in breast imaging.* Radiology, 1997. **205**: p. 399-406.
[47] Sompel, D.V.d., Brady, S.M., and Boone, J., *Task-based performance analysis of FBP, SART and ML for digital breast tomosynthesis using signal CNR and Channelised Hotelling Observers.* Medical Image Analysis, 2011. **15**(1): p. 53-70.
[48] Poplack, S.P., et al., *Digital breast tomosynthesis: Initial experience in 98 women with abnormal digital screening mammography.* American Journal of Roentgenology, 2007. **189**(3): p. 616-623.
[49] Yang, G., et al., *Design and feasibility studies of a stationary digital breast tomosynthesis system.* Nuclear Instruments and Methods in Physics Research A, 2011. **648**: p. 220-223.
[50] Zhou, W., et al., *Breast tomosynthesis reconstruction with a multi-beam x-ray source.* Proc. SPIE, 2009. **7258**.

[51] Zeng, G.L., *Medical Image Reconstruction: A Conceptual Tutorial*. 2009, Salt Lake City: Springer.
[52] Kak, A.C. and Slaney, M., *Principles of Computerized Tomographic Imaging*. 1988: The Institute of Ecletrical and Electronic Engineers.
[53] Zeng, G.L., *Image reconstruction: a tutorial.* Computerized Medical Imaging and Graphics, 2001. **25**: p. 97-103.
[54] S. Vandenberghe, Y.D.A., R. Van de Walle, T. Kauppinen, M. Koole, L. Bouwens, K. Van Laere, I. Lemahieu, R.A. Dierck, *Iterative reconstruction algorithms in nuclear medicine.* Computerized Medical Imaging and Graphics, 2011. **25**: p. 105-111.
[55] Bruyant, P.P., *Analytic and iterative reconstruction algorithms in SPECT.* Journal of Nuclear Medicine, 2002. **43**(10): p. 1343-58.
[56] Lee, N.-Y. and Choi, Y., *A modified OSEM algorithm for PET reconstruction using wavelet processing.* Computer Methods and Programs in Biomedicine, 2005. **80**: p. 236-245.
[57] Wang, C.X., et al., *Performance evaluation of filltered backprojection reconstruction and iterative reconstruction methods for PET images.* Computers in Biology and Medicine, 1998. **28**: p. 13-25.
[58] Renker, M., et al., *Iterative image reconstruction techniques: Applications for cardiac CT.* J Cardiovasc Comput Tomogr, 2011. **5**(4): p. 225-30.
[59] Ghetti, C., Ortenzia, O., and Serreli, G., *CT iterative reconstruction in image space: A phantom study.* Phys Med, 2011.
[60] Xu, F., *Fast Implementation of Iterative Reconstruction with Exact Ray-Driven Projector on GPUs.* Tsinghua Science and Technology, 2010. **15**(1): p. 30-35.
[61] Dobbins, J.T. and Godfrey, D.J., *Digital x-ray tomosynthesis: current state of the art and clinical potential.* Physics in Medicine and Biology, 2003. **48**: p. 65-106.
[62] Gordon, R., Bender, R., and Herman, G.T., *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography.* J Theor Biol, 1970. **29**(3): p. 471-81.
[63] Dempster, A.P., Laird, N.M., and Rubin, D.B., *Maximum Likelihood from Incomplete Data via the EM Algorithm.* Journal of the Royal Statistical Society. Series B, 1977. **39**(1): p. 1-38.
[64] Hudson, H.M. and Larkin, R.S., *Accelerated Image Reconstruction Using Ordered Subsets of Projection Data.* Transactions on Medical Imaging 1994. **13**(4): p. 601-609.
[65] Vintache, D., Humbert, B., and Brasse, D., *Iterative Reconstruction for Transmission Tomography on GPU Using Nvidia CUDA.* Tsinghua Science and Technology, 2010. **15**(1): p. 11-16.
[66] Nguyen, V.G., Lee, S.J., and Lee, M.N., *GPU-accelerated 3D Bayesian image reconstruction from Compton scattered data.* Physics in Medicine and Biology, 2011. **56**(9): p. 2817-36.
[67] Lee, S.J., Nguyen, V.G., and Lee, M.N., *Rapid 3-D Regularized EM Reconstruction for Compton Cameras Using GPU.* Medical Imaging 2010: Physics of Medical Imaging, 2010. **7622**.
[68] Chidlow, K. and Möller, T., *Rapid Emission Tomography Reconstruction.* Workshop on Volume Graphics, 2003.
[69] Xu, F. and Mueller, K., *Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware.* IEEE Trans Nucl Sci, 2005. **52**(3): p. 654–663.
[70] Goddard, I., et al., *Implementing an iterative reconstruction algorithm for digital breast tomosynthesis on graphics processing hardware - art. no. 61424V.* Medical Imaging 2006: Physics of Medical Imaging, Pts 1-3, 2006. **6142**: p. V1424-V1424.
[71] Pratx, G., et al., *Fast, Accurate and Shift-Varying Line Projections for Iterative Reconstruction Using the GPU.* Ieee Transactions on Medical Imaging, 2009. **28**(3).
[72] Lewitt, R.M. and Matej, S., *Overview of methods for image reconstruction from projections in emission computed tomography.* Proceedings of the Ieee, 2003. **91**(10): p. 1588-1611.
[73] Shepp, L.A. and Vardi, Y., *Maximum Likelihood Reconstruction for Emission Tomography.* IEEE Transactions on Medical Imaging, 1982. **MI-1**(2): p. 113-122.
[74] Lange, K. and Carson, R., *Em Reconstruction Algorithms for Emission and Transmission Tomography.* Journal of Computer Assisted Tomography, 1984. **8**(2): p. 306-316.

[75] Browne, J.A. and Holmes, T.J., *Developments with Maximum-Likelihood X-Ray Computed-Tomography.* Ieee Transactions on Medical Imaging, 1992. **11**(1): p. 40-52.
[76] Natterer, F. and Wübbeling, F., *Mathematical Methods in Image Reconstruction*, ed. J.E. Flaherty. 2001: Society for Industrial and Applied Mathematics.
[77] Moore, G.E., *Cramming more components onto integrated circuits.* Electronics, 1965. **38**(8).
[78] Kirk, D.B. and Hwu, W.-m.W., *Programming Massively Parallel Processors: A Hands-on Approach*. 2010: Morgan Kaufmann.
[79] Owens, J.D., et al., *Graphics Processing Units - powerful, programmable, and highly parallel - are increasingly targeting general-purpose computing applications.* Proceedings of the IEEE, 2008. **96**(5).
[80] Jang, B., et al., *Multi Gpu Implementation of Iterative Tomographic Reconstruction Algorithms.* ISBI'09 Proceedings of the Sixth IEEE international conference on Symposium on Biomedical Imaging: From Nano to Macro, 2009: p. 185-188.
[81] Castaño-Díez, D., et al., *Performance evaluation of image processing algorithms on the GPU.* Journal of Structural Biology, 2008. **164**: p. 153-160.
[82] Amdahl, G., *Validity of the single processor approach to achieving large scale computing capabilities*, in *AFIPS spring joint computer conference*. 1967: IBM Sunnyvale, California.
[83] NVIDIA, *NVIDIA CUDA C Programming Guide*. 2011.
[84] Gammex. *Mammographic Accreditation Phantom*. [acedido em 16 de Setembro de 2011]; Dísponivel em: http://www.gammex.com/n-portfolio/productpage.asp?id=299&category=Mammography&name=Mammographic+Accreditation+Phantom%2C+Gammex+156.
[85] Brown, B., et al. *GPU Acceleration of Iterative Digital Breast Tomosynthesis with Error Checking*. 2010 [acedido em 11 de Agosto de 2011]; Dísponivel em: http://www.gpucomputing.net/?q=node/407.
[86] NVIDIA. *CUDA Toolkit 3.2 Downloads*. 2011 [acedido em 27 de Fevereiro de 2011].
[87] Thrust. *Thrust*. 2011 [acedido em 25 de Março de 2011].
[88] Patel, S.J. and Stratton, J. *ECE 498 AL : Applied Parallel Programming*. 2010 [acedido em 11 de Agosto de 2011].
[89] Blelloch, G.E., *Scans as Primitive Parallel Operations.* IEEE Transactions on Computers, 1989. **38**(11): p. 1526-1538.
[90] Blelloch, G.E., *Prefix Sums and Their Applications*. 1990, Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University.
[91] Nguyen, H., *GPU Gems 3*. 2007: Addison-Wesley Professional.
[92] Siddon, R.L., *Fast calculation of the exact radiological path for a three-dimensional CT array.* Medical Physics, 1984. **12**(2): p. 252-255.
[93] Tingberg, A., *X-ray tomosynthesis: a review of its use for breast and chest imaging.* Radiation Protection Dosimetry, 2010. **139**(1-3): p. 100-107.
[94] Motta, A., et al., *Fast 3D-EM reconstruction using Planograms for stationary planar positron emission mammography camera.* Computerized Medical Imaging and Graphics, 2005. **29**(8): p. 587-596.
[95] Bushberg, J.T., *The Essential Physics of Medical Imaging*. 2002: Lippincott Williams & Wilkins.

ANEXO I Pseudo-Código do Programa desenvolvido

```
for each OSEM_iteration
        inicializar variaveis // Tamanho de bin, dimensões da FOV, imagem inicializada a
1(apenas para a primeira iteração)
        for each OSEM_subiteration
                ler os ficheiros de entrada com os dados das projecções
                determinar as coordenadas da fonte de raios-X
                        for each bloco_de_data //percorrer os blocos de dados com as dimensões
88x56 para metade do detector (aproveita a simetria)
                                atribuir a cada thread um bin do detector //cada thread é
identificada por um indice bidimensional correspondendo dessa forma a um bin localizado num
determinado ponto do detector
                                //obter as intersecções da LOR definida por cada bin e a fonte
de raios-X, com o eixo dos x:
                                Slopevectorx = xfocus-(thread_index_x+0.5)*xbinsize
                                Slopevectory = yfocus-(thread_index_y+0.5)*ybinsize
                                Slopevectorz = zfocus-(distancia_do_detector_ao_prato_de_apoio)

                                for each x_inteiro(thread_index_x+1 -> detector_x_dim/2)
                                        x = x_inteiro*xbinsize
                                        t = (x-(thread_index_x+0.5)*xbinsize) / Slopvectorx
                                        y = (thread_index_y+0.5)*ybinsize + t*Slopevectory
                                        z = distancia_do_detector_ao_prato_de_apoio
+t*Slopevectorz
                                        adicionar as coordenadas das intersecções x,y,z e o bin
a um array (um array para cada uma das coordenadas)
                                for each intersecção
                                        if(x_interseccao <0 OR x_interseccao > detector_x_dim/2
                                        OR y_interseccao <0 OR y_interseccao > detector_y_dim
                                        OR z_interseccao <0 OR z_interseccao > NSlices)
                                        then excluir intersecção
                                        //NSlices corresponde ao número de cortes obtidos como
pode ser observado na Figura 3.2
                                //achar as intersecções para os eixos y e z de forma análoga
                                agrupar as intersecções obtidas com os diferentes eixos
                                ordenar as intersecções usando a coordenada x e posteriormente
pelo bin
                                for each intersecção (intersecção+1 -> numero_total_intersecoes)
                                        //achar a distancia euclidiana entre cada intersecção:
                                        distancia = || intersection - intersection-1||
                                        //através das coordenadas das intersecções ver o voxel
correspondente à distancia calculada:
                                        if (x_interseccao == x_inteiro) //considerando que
x_inteiro se refere à escala normal, que não foi usada no trabalho
                                                then Coordenada_voxel_x = x-1
                                        else
                                                Coordenada_voxel_x = floor(x)
                                        //procedimento análogo para z
                                        if(y_interseccao < yfocus)
                                                if (y_interseccao == y_inteiro)
                                                        then Coordenada_voxel_y = y-1
                                                else
                                                        Coordenada_voxel_y = floor(y)
                                        else
                                                Coordenada_voxel_y = floor(y)
                                //após este passo obtém-se a matriz de sistema, composta por 3
arrays: bin, voxel e distancias
                                normalizar as distancias da matriz de sistema
                                //correr o algoritmo de reconstrução para metade do detector:
                                multiplicar a probabilidade de determinado voxel, com o
coeficiente de atenuação obtido na estimativa anterior //(1)
                                somar o resultado da multiplicação anterior para cada bin //(2)
                                dividir o input da projecção em cada bin pelo resultado da soma
anterior //(3)
                                ordenar a matriz de sistema, em ordem ao indice do voxel //(4)
                                multiplicação das probabilidades com o resultado de (3) //(5)
                                somar o resultado da multiplicação anterior para cada voxel e
somar à variavel de actualizacao //(6)
```

```
                                somar as probabilidades para cada voxel e somar à variavel de
normalizacao//(7)


                                inverter os dados da matriz de sistema através da simetria em x.
                                //correr o algoritmo de reconstrução outr vez com um passo
adicional aos 7 referidos anteriormente:
                                fazer update da imagem no caso de se estar na última iteração
dos blocos de dados.
                                Neste caso tem que se fazer reset às variaveis de actualizaçao e
normalizaçao //(8)

                                gravação da imagem
```