



Pedro Rafael Tomé Ferreira

Licenciatura em Ciências de Engenharia Biomédica

Optimization of Breast Tomosynthesis Image Reconstruction using Parallel Computing

Dissertação para obtenção do Grau de Mestre em
Engenharia Biomédica

Orientador : Nuno Matela, Prof. Auxiliar, FCUL

Co-orientadores : Pedro Medeiros, Prof. Associado, FCT/UNL
Nuno Oliveira, Investigador, IBEB

Júri:

Presidente: Prof. Doutora Carla Quintão Pereira

Arguente: Prof. Doutor André Damas Mora

Vogal: Prof. Doutor Nuno Matela



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Optimization of Breast Tomosynthesis Image Reconstruction using Parallel Computing

Copyright © Pedro Rafael Tomé Ferreira, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Acknowledgements

First of all, I would like to express my deep gratitude to my research supervisors: Professor Nuno Matela, for introducing me to this challenging, but rewarding, project, for all the time spent with my doubts and for his amazing knowledge in medical imaging; Professor Pedro Medeiros, for accepting to supervise me with this project, for guiding me through the defiant world of parallel programming using GPUs and for all the productive meetings we had along this dissertation; Nuno Oliveira, for his great knowledge of IDL, which demonstrated to be of great value to this dissertation and for his advices and suggestions.

I would like to show my appreciation for Professor Pedro Almeida, director of the Instituto de Biofísica e Engenharia Biomédica (Institute of Biophysics and Biomedical Engineering, IBEB), where this dissertation was conducted; I felt like I was home, always assisted by everyone with all the working conditions I needed. My special thanks are extended to the staff of IBEB for providing such a nice work environment.

I wish to express my recognition for everyone who works or has worked on this project. This dissertation is part of a large-scale research project in which several people developed their work in straight collaboration. Without this collaboration this work would not be as meaningful as it is.

I would like to show my appreciation for all the Professors from my faculty (Faculdade de Ciências e Tecnologia of the Universidade Nova de Lisboa, FCT-UNL) who made me the Engineer I am today. Particularly, I would like to show my tremendous respect to Professor Mário Secca, he was a cornerstone for me as a Biomedical Engineering student; thank you for your dedication to this program. I have also to thank the Professors of the Department of Informatics of the FCT-UNL due to their acceptance of a non-computer scientist among them asking questions about parallel programming, particularly to Professor Maria Cecília Gomes for letting me assist her classes of "High Performance Computing" which revealed to be of high-value for this dissertation. From a more general perspective, I have to thank the University of Twente, for the one-year Erasmus experience which greatly contributed to my broad academic background, and to Professor Wen-Mei Whu from the University of Illinois, for his coursera class about "Heterogenous Parallel

Programming", which I consider the best first introduction anyone can have to this topic, and for the productive conversations we had in Barcelona during the fifth edition of the Programming and Tuning Massively Parallel Systems summer school.

I am particularly grateful for the grant from the Department of Informatics of FCT-UNL, which supported this work besides letting me have my first teaching experience. I would like to thank particularly to: Professor Luis Caires, head of the department, for considering me to this grant; to the other teachers of the "Object-Oriented Programming" course and to my students, that without knowing gave me strengths to keep focused on this work.

To Ricardo Eleutério, this work would not be like this without your support, I am grateful for your companionship, friendship, patience, the long sleepless nights, the "keep up" spirit, wise advices and so on... I sincerely have a lot to thank to my other Biomedical Engineering colleagues, mainly Ana Carolina Pádua, Guilherme Coutinho, José Trindade and Margarida Félix, and to the persons I have met due to my extra-curricular courses from Computer Science, specially Farah Mussa and Jacomina Guerreiro.

To my long-date friends, specially Diana Matos and Raquel Santos: I am grateful to have met you during my lifetime, each of you made me who I am today.

For everything my family have sacrificed to offer me a high-level education, thank you. I hope that I have lived to the expectations and I am proud to be part of this family. To my dad, that God has his soul, to my mother, which is the best and most kind person I know, to my brother and sister-in-law, their support were essential, to my aunt Isaura, for her affection, ... Without their love and support this effort would not mean the same...

Finally, thank you to you, the reader, for being interested in this work!

Abstract

Breast cancer is the most common cancer among women, being a major public health problem. Worldwide, X-ray mammography is the current gold-standard for medical imaging of breast cancer. However, it has associated some well-known limitations. The false-negative rates, up to 66% in symptomatic women, and the false-positive rates, up to 60%, are a continued source of concern and debate. These drawbacks prompt the development of other imaging techniques for breast cancer detection, in which Digital Breast Tomosynthesis (DBT) is included. DBT is a 3D radiographic technique that reduces the obscuring effect of tissue overlap and appears to address both issues of false-negative and false-positive rates. The 3D images in DBT are only achieved through image reconstruction methods. These methods play an important role in a clinical setting since there is a need to implement a reconstruction process that is both accurate and fast.

This dissertation deals with the optimization of iterative algorithms, with parallel computing through an implementation on Graphics Processing Units (GPUs) to make the 3D reconstruction faster using Compute Unified Device Architecture (CUDA). Iterative algorithms have shown to produce the highest quality DBT images, but since they are computationally intensive, their clinical use is currently rejected. These algorithms have the potential to reduce patient dose in DBT scans.

A method of integrating CUDA in Interactive Data Language (IDL) is proposed in order to accelerate the DBT image reconstructions. This method has never been attempted before for DBT. In this work the system matrix calculation, the most computationally expensive part of iterative algorithms, is accelerated. A speedup of 1.6 is achieved proving the fact that GPUs can accelerate the IDL implementation.

Keywords: Digital Breast Tomosynthesis, Iterative Image Reconstruction, Parallel Programming, GPU, CUDA, IDL

Resumo

O cancro da mama é o tipo de cancro mais comum entre as mulheres, sendo um importante problema de saúde pública. Em todo o mundo, a mamografia de raios-X é o *gold-standard* da imagiologia de cancro da mama. No entanto, tem associada algumas limitações bem conhecidas. A taxa de falso-negativos, até 66% em mulheres sintomáticas, e a taxa de falso-positivos, até 60%, são uma fonte constante de preocupação e debate. Estas desvantagens propiciaram o desenvolvimento de outras técnicas de imagem médica para a deteção de cancro da mama, no qual a tomossíntese aplicada à mama (DBT, do inglês *Digital Breast Tomosynthesis*) está incluída. A DBT é uma técnica radiográfica 3D que reduz o efeito de obscurecimento da sobreposição de tecidos e parece resolver ambas as questões das taxas de falsos negativos e falsos positivos. As imagens 3D em DBT apenas são obtidas através de métodos de reconstrução de imagem. Estes métodos têm um papel importante num contexto clínico uma vez que há a necessidade de implementar um processo de reconstrução que seja simultaneamente preciso e rápido.

Esta dissertação propõe a otimização de algoritmos iterativos, com computação paralela através da implementação em unidades de processamento gráfico (GPUs) para tornar a reconstrução 3D mais rápida usando *Compute Unified Device Architecture* (CUDA). Algoritmos iterativos têm demonstrado produzir imagens de DBT da melhor qualidade mas, uma vez que são computacionalmente exigentes, o seu uso clínico é atualmente rejeitado. Estes algoritmos têm o potencial de reduzir a dose do paciente em exames de DBT.

Um método de integração de CUDA em *Interactive Data Language* (IDL) é proposto no sentido de acelerar a reconstrução de imagens de DBT. Este método nunca antes foi tentado para DBT. Neste trabalho o cálculo da matriz do sistema, a parte computacionalmente mais exigente dos algoritmos iterativos, é acelerado. Um *speedup* de 1,6 é conseguido provando o fato de que GPUs aceleram a implementação em IDL.

Palavras-chave: Tomossíntese para Mamografia, Reconstrução de Imagem Iterativa, Programação Paralela, GPU, CUDA, IDL

Contents

List of Figures	xiii
List of Tables	xvii
Listings	xix
Acronyms and Abbreviations	xxi
1 Introduction	1
1.1 Objectives	3
1.2 Dissertation Overview	3
2 Background	5
2.1 Physiological Background	5
2.1.1 Outline of the Anatomy and Physiology of the Breast	5
2.1.2 Breast Cancer	6
2.2 Breast Cancer Imaging	6
2.2.1 Breast Tomosynthesis	7
2.3 Image Reconstruction in Tomosynthesis	10
2.3.1 Simultaneous Algebraic Reconstruction Technique	13
2.3.2 Maximum Likelihood – Expectation Maximization	14
2.3.3 Ordered Subsets – Expectation Maximization	15
2.4 Parallel Programming	15
2.4.1 GPU	16
2.4.2 CUDA	18
2.4.3 State of the Art of Image Reconstruction using Parallel Programming	20
3 Materials and Methods	21
3.1 DBT System	21
3.2 IDL Implementation	23

3.2.1	System Matrix Calculation	25
3.3	Fundamentals of CUDA Programming	29
3.3.1	GPU-Accelerated Libraries	32
3.3.1.1	Thrust	33
3.3.2	Development Tools	34
3.4	CUDA Integration in IDL	35
3.4.1	Hardware & Software	36
3.4.2	CUDA Code	37
3.4.3	Invoke CUDA from IDL	39
3.4.4	Evaluation of the CUDA-IDL Implementation	41
4	Results and Discussion	43
4.1	Expended Time in the System Matrix Calculation	43
4.2	Single-Thread Execution per Kernel Strategy	44
4.3	Full Integration of CUDA in IDL	47
5	Conclusions and Future Work	51
5.1	Contributions	54
A	Poster of WBME	69
B	Poster of PUMPS	71
C	Quick Guide	73

List of Figures

1.1	Most common cancer types for women in the United States, 2014. (a) Estimated new cases. (b) Estimated number of deaths. Adapted from [1]. . . .	1
2.1	Anatomy of the female breast [54].	6
2.2	Basic principles of breast tomosynthesis. Image data are acquired from various angles as the X-ray tube moves according to the (a) step-and-shoot method or the (b) continuous exposure method. (c) Image data acquired from different angles. Further, these images are reconstructed to provide a 3D image of the breast where the two overlapping structures would be located in different planes. Adapted from [42].	8
2.3	Tissue overlap in X-ray mammography (2D), which hides pathologies (pink lesion). With digital breast tomosynthesis (3D), the cross-sectional slices make the lesion less likely to be obscured [72].	9
2.4	Schematic of iterative image reconstruction algorithm for DBT.	13
2.5	Execution of a CUDA program, using a multidimensional example of a CUDA grid organization. Adapted from [121].	17
2.6	CUDA memory model. Adapted from [121].	18
2.7	An example of barrier synchronization [121].	19
3.1	Siemens MAMMOMAT Inspiration [135].	22

3.2	Diagram of the Siemens MAMMOMAT Inspiration system performing DBT. An X-ray source covers a 50° arc (-25° to $+25^\circ$, θ) emitting low-dose X-rays at 25 locations (in this image is only presented 15 locations for simplicity). Twenty five corresponding projections are acquired by the detector. The most relevant dimensions of the system are: 65.5 cm between the X-ray tube in the initial state (when $\theta = 0^\circ$) and the breast support table, 3 cm between the axis of rotation and the breast support table and 1.7 cm between the breast support table and the detector. The compressed breast in this work is 6 cm and it is related with the number of slices in the 3D reconstructed image (N_{slices}). y_{focus} and z_{focus} represent, respectively, the coordinates y and z of the detector, both depending on θ . Adapted from [133].	23
3.3	Schematic of the iterative image reconstruction algorithms for DBT applied to the IDL implementation.	24
3.4	Two-dimensional example on the xy plane of the ray driven approach. The \overrightarrow{AB} ray intersects the horizontal (filled circles) and vertical (open circles) lines. d_5 represents the 5^{th} intersection length of the ray within the image — Euclidean distance between the points with x -coordinate X_2 and X_3 . The extension to three-dimensional is straightforward.	25
3.5	Illustration of a ray i (R_i) from a detector bin i to the X-ray beam focus, illustrating that R_i is attenuated only by voxels that are crossed by it (green voxels).	26
3.6	Focus positions on (a) xy and (b) yz planes.	27
3.7	Matching of the intersection points of a ray and the voxels on the (a) xz plane (which has always a positive slope) and (b) yz plane (which in this case has a negative slope).	28
3.8	Execution of an IDL program invoking CUDA and using a multidimensional example of a CUDA grid organization.	36
3.9	Illustration of how the data received in IDL from the CUDA code (the voxels' coordinates and distances) are processed to organize it per bin.	40
4.1	Reconstructed images of the 27^{th} z -plane for a single iteration for System 1 with a bins' scale factor of 16 using (a)–(c) SART, (d)–(f) ML-EM and (g)–(i) OS-EM. Three different images per algorithm are shown, from left to right: CUDA-IDL implementation using a single-thread execution per kernel, pure-IDL implementation and difference between both.	46
4.2	The NME percentages of the 3D DBT reconstructed image with OS-EM as a function of the number of subsets for a single iteration for System 1 with a bins' scale factor of 16.	47

4.3	Reconstructed images of the 27 th z -plane for a single iteration for System 1 with a bins' scale factor of 16 using (a)–(c) SART and (d)–(f) ML-EM. Three different images per algorithm are shown, from left to right: CUDA-IDL implementation using a multi-thread scheme per kernel call, pure-IDL implementation and difference between both.	50
5.1	Larger project entitled "Improvements of image quality and dose reduction in digital breast tomosynthesis using statistical image reconstruction algorithms" in which this dissertation is integrated.	52

List of Tables

3.1	Functions used on the host and device to allocate and free memory in CUDA.	30
3.2	Systems used in this work.	36
3.3	Devices used in this work.	36
4.1	Percentages of the entire computation of a single iteration spent in the calculation of the system matrix. These values are obtained from the pure-IDL implementation performed in two different systems for SART, ML-EM and OS-EM iterative algorithms, with three different bins' scale factors. Each presented value is an average of five different tests. The standard deviation of each value is not higher than 0.2%.	44
4.2	NME percentages of a single iteration obtained through the comparison of the pure-IDL implementation with the CUDA-IDL implementation using a single thread CUDA kernel. This test was performed in two different systems for SART, ML-EM and OS-EM iterative algorithms and with a bins' scale factor of 16.	45
4.3	Average of five execution times of the pure-IDL and the CUDA-IDL implementations and resulting speedup. These tests were completed for a single iteration of SART and ML-EM for the two available systems and for a bins' scale factor of 16. The standard deviation of both implementations do not exceed 4 s and 0.04 regarding the execution times and the speedups, respectively.	48
4.4	NME percentages of a single iteration obtained through the comparison of the pure-IDL implementation with the CUDA-IDL implementation using the standard multiple thread kernel. This test was performed in two different systems for SART and ML-EM iterative algorithms and with a bins' scale factor of 16.	49

Listings

3.1	Vector Addition — C Code.	30
3.2	Vector Addition — CUDA Code.	31
3.3	<code>remove_copy_if</code> — Thrust function example.	34
3.4	<code>sort_by_key</code> — Thrust function example.	34
3.5	Pseudocode of the CUDA code for the system matrix calculation.	38

Acronyms and Abbreviations

API	Application Programming Interface
ART	Algebraic Reconstruction Technique
CPU	Central Processing Unit
CT	Computed Tomography
CUDA	Compute Unified Device Architecture
DBT	Digital Breast Tomosynthesis
DCIS	Ductal Carcinoma <i>In Situ</i>
DM	Digital Mammography
FBP	Filtered BackProjection
FDA	Food and Drug Administration
FOV	Field Of View
GDL	GNU Data Language
GPU	Graphics Processing Unit
IDC	Invasive Ductal Carcinoma
IDL	Interactive Data Language
ILC	Invasive Lobular Carcinoma
LCIS	Lobular Carcinoma <i>In Situ</i>
ML-EM	Maximum Likelihood – Expectation Maximization
MRI	Magnetic Resonance Imaging

NME Normalized Mean Error

NVCC NVIDIA's CUDA Compiler

OS-EM Ordered Subsets – Expectation Maximization

PET Positron Emission Tomography

SAA Shift-And-Add

SART Simultaneous Algebraic Reconstruction Technique

SIRT Simultaneous Iterative Reconstruction Technique

SNR Signal-to-Noise Ratio

SSH Secure Shell

STL Standard Template Library

TV Total Variation

X11 X Window System, Version 11

Introduction

Breast cancer is the most common cancer among women, being a major public health problem. In the United States it is projected [1] that a total of 232,670 new cases and 40,000 deaths will occur in 2014 due to breast cancer (Figure 1.1). In Portugal, breast cancer in women is the leading type of cancer, both in terms of new cases and deaths, 6,090 and 1,570 in 2012 respectively [2]. In fact, all countries in Europe showed in 2012 that breast is the most common site for cancer in women [2]. In Portugal alone, the statistics show that one in eleven women are likely to develop breast cancer throughout their lives.

There are several risk factors that are already well-established [3] for the development of breast cancer: age [4], family history [5], [6], early menarche [7]–[9], late menopause [10],

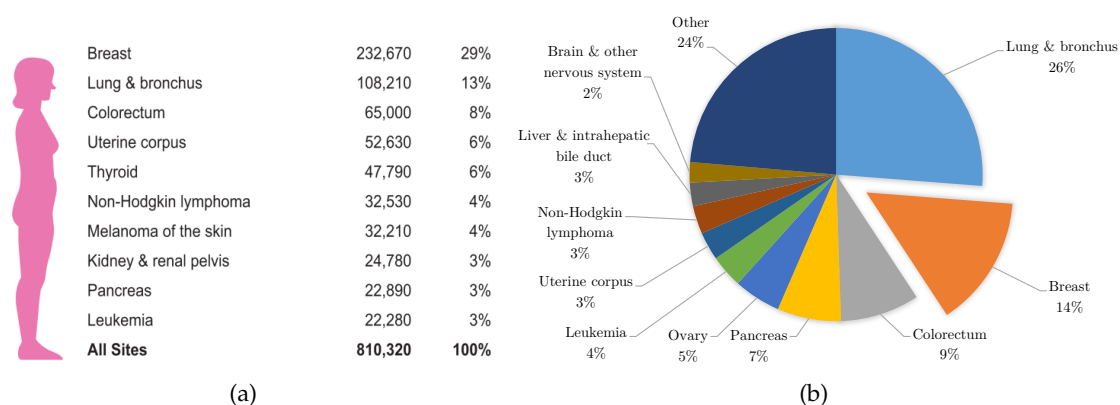


Figure 1.1: Most common cancer types for women in the United States, 2014. (a) Estimated new cases. (b) Estimated number of deaths. Adapted from [1].

nulliparity [11], late age at first full-term pregnancy [12] and use of hormone replacement therapy [13]. Other risk factors include: geographic location and socioeconomic status [14]–[17], ionizing radiation [18]–[20], diet [21]–[24], alcohol consumption [25]–[27] and body weight [28]–[31].

In the 1990s, the rising trend of death rates due to breast cancer in women has changed in the United States and in most European countries (including Portugal). Over the past 25 years the death rates have been decreasing [32]–[34] as a consequence of improved treatment and early detection through screening and increasing awareness [35].

All over the world, X-ray mammography is the current gold-standard for medical imaging of breast cancer [36]. However, it has associated some well-known limitations. The false-negative rates, up to 66% in symptomatic women [37]–[39], and the false-positive rates, up to 60% [40], [41], are a continued source of concern and debate. These drawbacks prompt the development of other imaging techniques for breast cancer detection, in which Digital Breast Tomosynthesis (DBT) is included. DBT is a 3D radiographic technique that reduces the obscuring effect of tissue overlap and appears to address both issues of false-negative and false-positive rates [42], [43]. Besides being a new technique that was recently approved by the United States Food and Drug Administration (FDA) [44], studies [45]–[47] have been showing that it is subjectively better than conventional mammography in highlighting masses and areas of architectural distortion. Therefore, tomosynthesis has great potential to be a valuable tool in routine screening for breast cancer. In fact, the acquisition is performed with the same equipment as digital mammography and it is even possible to obtain a 2D mammography image from tomosynthesis images without the need of an extra acquisition.

The 3D images in DBT are only achieved through reconstruction methods. These methods play an important role in a clinical setting since there is a need to implement a reconstruction process that is both accurate and fast. There are two major groups of reconstruction algorithms, the analytical and the iterative. The former include Filtered BackProjection (FBP) and the latter the Simultaneous Algebraic Reconstruction Technique (SART), the Maximum Likelihood – Expectation Maximization (ML-EM) and the Ordered Subsets – Expectation Maximization (OS-EM) reconstruction algorithms. In analytical algorithms, such as FBP, analytical models that simplify reconstructions are employed [48]. On the other hand, iterative algorithms use geometric models with much more precision, producing higher-quality images, but are much more computationally intensive [49]–[51]. Therefore, the application of these reconstruction methods in clinical practice requires performance optimization, which can be achieved via parallel computing by implementation on platforms such as Graphics Processing Units (GPUs) to make the 3D reconstruction faster. In fact, with the recent progress in this type of hardware, GPUs have the potential to change the picture of medical image reconstruction algorithms in a very dramatic way.

1.1 Objectives

This work is part of a large-scale two-year research project which is concluding its first year. This larger project is entitled "Improvements of image quality and dose reduction in digital breast tomosynthesis using statistical image reconstruction algorithms", is conducted in straight collaboration between multiple people from different institutions — Instituto de Biofísica e Engenharia Biomédica, Campus Tecnológico e Nuclear, Hospital da Luz and Laboratório de Instrumentação e Física Experimental de Partículas — and aims to:

- increase clinical value of tomosynthesis by improving image quality;
- decrease patient dose in DBT scans;
- develop a tool to optimise the exam regarding dose and image quality;
- create a new dosimetry system for mammography screening.

This dissertation, which was conducted at the Instituto de Biofísica e Engenharia Biomédica (Institute of Biophysics and Biomedical Engineering), proposes the development of a fast and accurate DBT-based image reconstruction method that includes the optimization of three iterative algorithms (SART, ML-EM and OS-EM) with parallel programming, using NVIDIA® CUDA®¹ to program GPUs. The time-optimization of such algorithms can be of great importance since if they have turnaround times compatible with their use in a real clinical setting they could reduce the radiation dose the patients receive per medical examination as suggested by some studies [52], [53]. This is particularly important since nowadays there is an increasing concern about reducing the radiation dose received by patients during a medical examination [20].

1.2 Dissertation Overview

This dissertation is organized in five main chapters, being the first one this introduction. The remainder of the dissertation is organised as follows: Chapter 2 details the background of the work developed in this dissertation, in particular the physiological background related to the breast and tumours, a brief review of the breast cancer imaging modalities (focusing on tomosynthesis), image reconstruction in tomosynthesis and finalizes with a first introduction to parallel programming. Chapter 3 provides the materials and the methodology followed throughout this work. Chapter 4 is devoted to summarizing the key results of this dissertation and Chapter 5 presents overall conclusions and perspectives of future work.

¹NVIDIA and CUDA are registered trademarks of NVIDIA Corporation in the United States and other countries.

2

Background

In this chapter the background information necessary for the understanding of this work is addressed. First the physiological background is discussed in Section 2.1, followed by the review of the different techniques to access breast cancer, focusing on DBT, in Section 2.2. The different algorithms for DBT image reconstruction are described in Section 2.3, and finally in Section 2.4 an introduction to parallel programming and a state of the art of image reconstruction through the use of parallel programming is provided.

2.1 Physiological Background

2.1.1 Outline of the Anatomy and Physiology of the Breast

In order to understand breast cancer, it helps to have some basic knowledge about the normal structure of the breast, shown in Figure 2.1. The female breast is mostly constituted by fibrous, glandular and fatty (adipose) tissue. As women become older, adipose tissue gradually replaces glandular tissue, decreasing the mammographic density [55]. The fibrous and glandular tissues are structured into ducts and lobules, which are surrounded by fat. Additionally, within the adipose tissue is a network of nerves, lymph vessels, lymph nodes, and blood vessels.

The lymphatic system is a network of lymphatic vessels and lymph nodes running throughout the entire body. Lymph nodes are bean-shaped collections [56] of immune system cells that are connected by lymphatic vessels. Clusters of lymph nodes are fixed in areas throughout the lymphatic system. Lymphatic vessels carry a clear fluid called lymph away from the breast, acting as filters. Lymph contains tissue fluid and waste

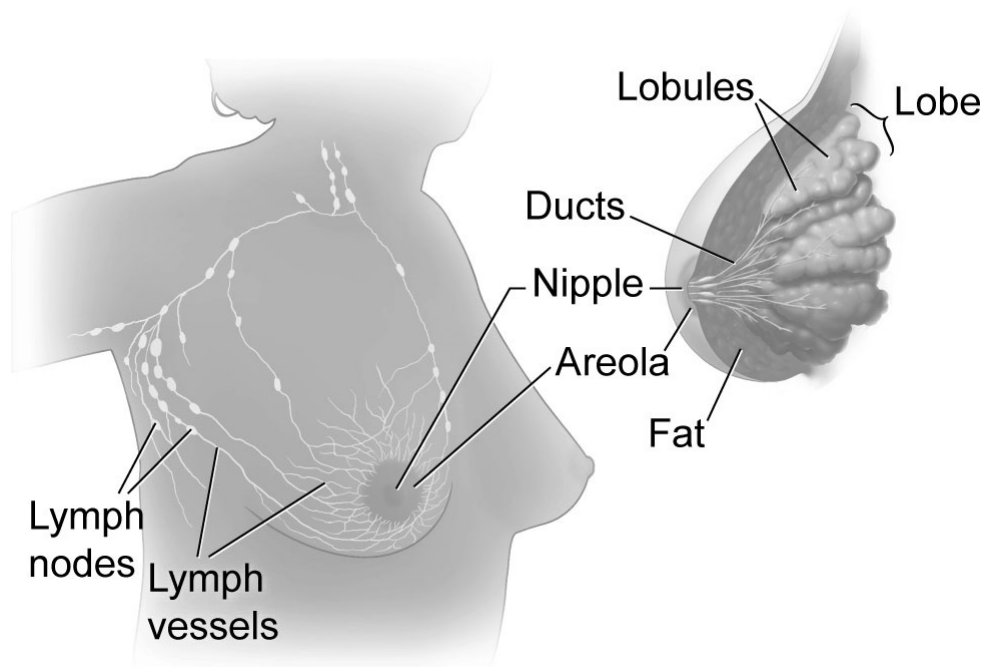


Figure 2.1: Anatomy of the female breast [54].

products, as well as immune system cells [56]. Most lymphatic vessels in the breast connect to lymph nodes under the arm (axillary nodes), being the first place a breast cancer will spread to. The understanding of the lymphatic system is therefore essential since breast cancer cells can enter lymphatic vessels and begin to grow in lymph nodes.

2.1.2 Breast Cancer

There are different types of breast cancer, being the most common those that affect the ducts or lobules (Figure 2.1). They can be *in situ*, within the terminal duct lobular unit and adjacent ducts, or invasive, when they penetrate the basement membrane [57]. The earliest form of breast cancer is Ductal Carcinoma *In Situ* (DCIS) [58], in which the cancer cells are confined to the breast ducts; Lobular Carcinoma *In Situ* (LCIS) is when it is confined to the lobules. Both DCIS and LCIS can degenerate into invasive carcinoma, also known as Invasive Ductal Carcinoma (IDC) and Invasive Lobular Carcinoma (ILC), respectively, in which IDC is the most common type of invasive breast cancer (approximately 75% [57]).

When a carcinoma is found in the breast there is a need to verify how far the cancer has spread. In order to do that, a sentinel lymph node biopsy can be performed, detecting if and which lymph nodes are cancerous.

2.2 Breast Cancer Imaging

There are two main diagnostic pathways [57]: screening and symptomatic. Breast screening will be covered in this section.

The current gold-standard for medical imaging of breast cancer [36] is the X-ray mammography despite its well-known limitations. However, due to the different types of cancer, each imaging technique presents a different ability of detecting tumours. Besides X-ray mammography, breast ultrasound and Magnetic Resonance Imaging (MRI) are also established diagnostic techniques.

Breast ultrasound is mainly used in follow-up examination when suspicious masses are initially detected with X-ray mammography — has the capacity of distinguishing tumours from cystic masses (fluid in nature and almost always benign) — and also for examination of a palpable breast mass in pregnant women and in women with dense breast tissue (usually younger females) [59]. Ultrasound is a low-cost non-ionising technique, but is a difficult technique to detect and characterise smaller tumours (due to image resolution), produces very noisy images, depends on the skills of the operator, and do not allow an accurate differentiation between benign and malign pathologies [60].

MRI is well-known for its high contrast and spatial resolution. Nevertheless, the false-positive rates can be a significant problem while interpreting a breast MRI [61]. The use of intravenous contrast, the actual lack of commercially available biopsy systems and its costs, are also some drawbacks of MRI.

Additionally, other techniques have also been investigated for breast imaging, such as scintimammography and Positron Emission Tomography (PET), both nuclear imaging modalities [62] that provide useful complementary information. However, since both these techniques are tuned for whole body imaging, they lack in the detection of small lesions in the breast.

Newer techniques that are now under investigation for breast imaging, where the tests are in the earliest stages of research, include optical imaging [63], [64], positron emission mammography [65]–[67] and microwave imaging [68]–[70].

The current limitations of the breast cancer imaging techniques led to the emergence of a new interest in research on breast tomosynthesis, being this a promising technique.

2.2.1 Breast Tomosynthesis

Systems for Digital Breast Tomosynthesis (DBT), commonly called 3D mammography, are now being used for clinical work in several parts of the world, including the United States, Canada, Asia and Europe [71]. Particularly, in Portugal seven systems for DBT are installed in some hospitals (e.g. Instituto Português de Oncologia de Lisboa and Hospital da Luz), three from Siemens^{®2} and four from Hologic^{®3}. DBT is a new emerging technique that was in 2011 approved by the United States FDA [44], besides being available for clinical use in several other countries since 2009 [72].

DBT is a method that performs 3D X-ray mammography at doses similar to conventional X-ray mammography, being the total radiation dose per acquisition approximately

²Siemens is a registered trademark of Siemens AG.

³Hologic is a trademark of Hologic, Inc., in the United States and other countries.

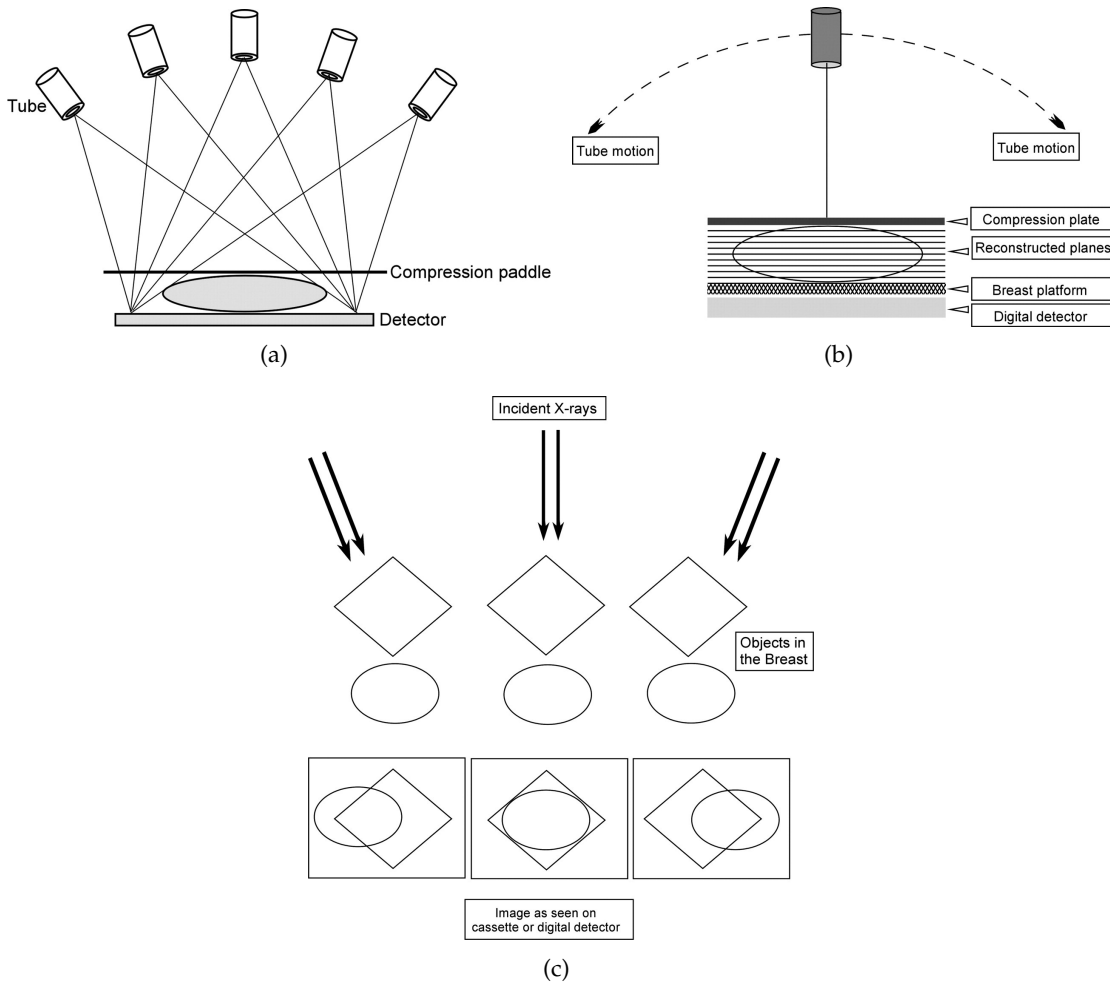


Figure 2.2: Basic principles of breast tomosynthesis. Image data are acquired from various angles as the X-ray tube moves according to the (a) step-and-shoot method or the (b) continuous exposure method. (c) Image data acquired from different angles. Further, these images are reconstructed to provide a 3D image of the breast where the two overlapping structures would be located in different planes. Adapted from [42].

twice the current dose per exposure of Digital Mammography (DM) [43], [73]. DBT acquisition is similar to conventional mammography with regard to breast positioning and compression; the difference is that the X-ray tube arcs over the breast taking multiple low-dose exposures (typically 9–25 [74]) that last several seconds (ranging from 4 to 25 seconds [74]) (Figure 2.2). This motion can be continuous (pulsed short exposures during continuous motion of the X-ray source) or performed with the step-and-shot method (one exposure at each position of the tube between movements). The all set of projections is then reconstructed by a computer algorithm, obtaining a 3D image of the breast on which one can analyse sections that usually have a thickness of one millimetre. In this way, it is possible to eliminate the biggest problem of X-ray mammography: the overlap of different structures [75]–[78]. With X-ray mammography small breast cancers may be hidden, resulting in a false-negative result (Figure 2.3) [37]–[39], or normal structures can blend

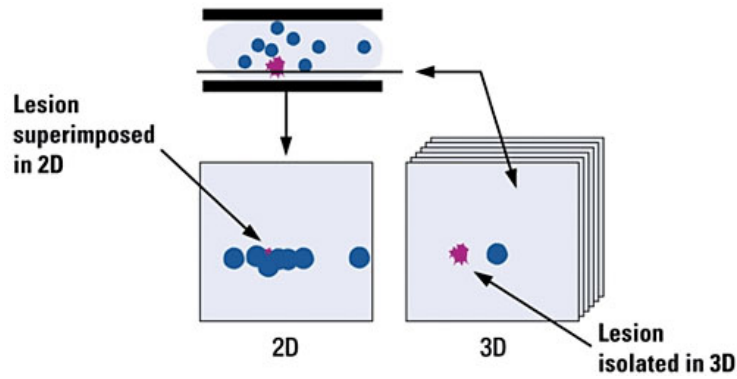


Figure 2.3: Tissue overlap in X-ray mammography (2D), which hides pathologies (pink lesion). With digital breast tomosynthesis (3D), the cross-sectional slices make the lesion less likely to be obscured [72].

together giving the appearance of pathology, resulting in a false-positive reading [40], [41] that leads to unnecessary recalls and biopsies. The overlapping occurs because the signal detected at a location on the detector is dependent upon the total attenuation of all the tissue between the detector and the X-ray beam. The attenuation can be represented by the Beer-Lambert law:

$$\frac{I}{I_0} = e^{-\mu x} \quad (2.1)$$

where I is the measured intensity, I_0 the incident intensity, x the thickness of the material the beam travels through and μ the linear attenuation coefficient, which is higher for denser tissues.

Early findings mention that DBT may replace X-ray mammography as the standard tool for screening [71] because of its improved effectiveness (sensitivity and specificity). DBT might be the greatest technological milestone in the field of breast diagnosis by mammography, since the invention of mammography. In particular, DBT may offer the following potential clinical benefits:

- Improved cancer detection (elimination of overlapping tissues) [79]–[81];
- Reduced recalls (fewer biopsies) [80]–[82];
- Less painful compression [83], [84];
- Clearer images [43], [47], [82], [85];
- Faster review and tissue localization (biopsy methods) [86]–[88];
- Improved radiologist confidence [89].

Despite its many advantages, DBT has limitations that should be taken into account. Potential disadvantages of DBT include: large number of reconstructed images to analyse per patient, no significant value is added in interpreting lesions that are well demonstrated with X-ray mammography, dense tissue within the plane of interest has the same

limiting effect as with X-ray mammography, additional radiation exposure if used as adjunct to X-ray mammography, image artefacts and the fact that the appearances of the parenchyma and normal structures on DBT images may vary from those on X-ray mammography. For this reason it is suggested [42] for radiologists to have additional training in order to interpret DBT images.

Nowadays, DBT still has several areas for improvement, including the distribution of radiation dose among the projections, the reconstruction algorithm, image processing and the overall user interface. The dose reduction from current levels is a potential benefit that is worth pursuing since it can facilitate the acceptance of tomosynthesis.

2.3 Image Reconstruction in Tomosynthesis

In DBT, as with Computed Tomography (CT), a 3D image is created from a sequence of projection views using reconstructed algorithms. This is an extremely challenging task since the image data are inherently incomplete, mainly with DBT — lower number of projections and limited angle in which the DBT system arcs around the breast.

Mathematically, the reconstruction problem can be thought of as an undetermined linear system. In fact, it is an inverse problem in which one uses the results from the projections to infer the value of the attenuation coefficient of each of the voxels in the 3D image — there will not be a unique solution (even with ideal data). The voxels determine how that portion of the image affects the X-ray intensity. In linear imaging problems, the following model is used:

$$Y = Ax + \eta \quad (2.2)$$

where Y corresponds to the measured data, A is the system matrix which gives the data measurement process in the image (geometric model of the transmission and detection processes), x is the image to be estimated and η is the additive noise (in DBT it can be for instance X-ray scatter and electronic noise), which is neglected in order to simplify the problem. However, for DBT there is multiple projections, therefore:

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_n \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} A_1 \\ \vdots \\ A_n \\ \vdots \\ A_N \end{bmatrix} x \quad (2.3)$$

being N the total number of projections and n a single projection such that $1 \leq n \leq N$.

Furthermore, considering that each projection has I pixels and that x has J voxels:

$$\begin{bmatrix} y_1 & \cdots & y_i & \cdots & y_I \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1I} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{iI} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{I1} & \cdots & a_{Ij} & \cdots & a_{II} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_I \end{bmatrix} \quad (2.4)$$

where y_i is the i^{th} pixel of Y_n , a_{ij} is the system matrix value for the combination of the pixel i and the voxel j and x_j is the j^{th} voxel of x . Finally, for a pixel i of a single projection:

$$y_i = \sum_j a_{ij} x_j \quad (2.5)$$

In order to obtain the attenuation coefficient for each voxel j , it is important to take into consideration the DBT system, the inhomogeneity of the breast and the Beer-Lambert law (Equation 2.1), that is:

$$\ln \left(\frac{I_0}{I} \right)_i = \sum_j \mu_j x_{ij} \quad (2.6)$$

for each projection, where: $\ln \left(\frac{I_0}{I} \right)_i$ represents the X-ray beam attenuation in the direction i , which is the direction defined by the focal point of the emission and the i^{th} bin of the detector (from now on this direction will be called ray i , R_i); μ_j is the attenuation coefficient in the voxel j and x_{ij} represents the probability that a photon in R_i has to be absorbed in the voxel j . From Equation 2.6 it is possible to conclude that the attenuation coefficient for a voxel j is not possible to identify with only one measurement, due to the high number of unknown values in the equation. The attenuation coefficients can only be separated with multiple projections. Through the assignment of gray levels to different ranges of attenuation coefficients, the 3D gray-scale image can be produced, representing the different structures in the breast with different X-ray attenuation characteristics.

Every image reconstruction algorithm should fulfil Equation 2.2. This equation can be adapted to DBT by comparing Equation 2.5 with Equation 2.6: $\ln \left(\frac{I_0}{I} \right)_i$ corresponds to y_i ; $\sum_j \mu_j x_{ij}$ corresponds to $\sum_j a_{ij} x_j$, where μ_j corresponds to x_j and x_{ij} to a_{ij} , depending the system matrix only on the scanner's geometry.

Reconstruction algorithms are a critical element for DBT as they affect the image quality. These algorithms can be divided, based on the mathematical approach, in two major groups: iterative and analytical. The focus of this dissertation will be the iterative algorithms, which can be classified as algebraic or statistical. In turn, statistical algorithms can be approached as having two different types which depend on the nature of the noise in the acquired data (the assumption of a Poisson distribution or the Gaussian noise). Analytical algorithms are based on an idealized mathematical model for the data, over-simplifying the physics inherent to the processes in DBT, which limits the accuracy

of the reconstructed images. Several analytic algorithms have been employed in DBT, including:

- **Shift-And-Add (SAA)** [90], [91] — algorithm based on shifting and adding projections to sharpen the plane focus. It is frequently used in DBT [73] due to its simplicity and speed, being the initial idea for DBT image reconstructions [92]. However, due to the characteristics of the reconstruction, a considerable amount of out of focus slice blur occurs. This algorithm has been modified by several research groups (e.g. image stretching shift-and-add [93]).
- **Filtered BackProjection (FBP)** [94]–[96] — most used and common known algorithm in medical image reconstruction [97]–[100], due to the fast execution time and easy implementation. Despite its efficiency, FBP tends to yield conspicuous artefacts [48], [99], [101]. Therefore, the limited number of projections in DBT makes the reconstruction with FBP inexact. FBP performs better with wider angles [74], [102].

Analytic algorithms present multiple problems that iterative algorithms can improve. The fact that iterative algorithms include accurate physical and statistical models usually results as an improved image accuracy, which is an advantage upon analytical algorithms that employ simplifications in the reconstruction model (e.g. unable to handle scatter) [48]. However, since iterative algorithms frequently have to solve large sets of nonlinear equations, they take much more time when compared to the analytical algorithms [49]–[51], which explains the default clinical choice of analytical algorithms. Nevertheless, with the recent improvements made in the computers' hardware field, iterative reconstruction algorithms must now be considered in order to improve image quality and, consequently, the diagnosis and treatment. In fact, with the rise of parallel computing in the medical physics field, it is now possible to accelerate several processes that could not be done a couple of years ago. Furthermore, by using iterative algorithms, the reduction of radiation dose that is induced to patients in these exams becomes a possibility [52], [53], since these algorithms deal better with noise than the analytical ones.

Generally, every iterative image reconstruction algorithm can be explained using the scheme presented in Figure 2.4 (being this scheme adapted to DBT). The reconstruction process begins with an initial image estimate (*Initialization*), $\hat{x}^{(0)}$, usually a constant. A *Forward Projection* operation — to estimate each bin value from the linear attenuation coefficients stored in voxels — is then applied to the current "Estimated 3D Image", $\hat{x}^{(k)}$, yielding a set of "Estimated Projections". After, a *Comparison* is completed between those projections and the "Original X-ray projections" (Y), obtaining "Projection Errors". Next, with the process of *Backprojection* — to update voxels based on differences between estimated and measured bin values — an "Errors Map" is obtained which is used to *Update* $\hat{x}^{(k)}$, obtaining $\hat{x}^{(k+1)}$. In both the *Forward Projection* and the *Backprojection* processes, the system matrix A is taken into account to model the radiation transmission and detection

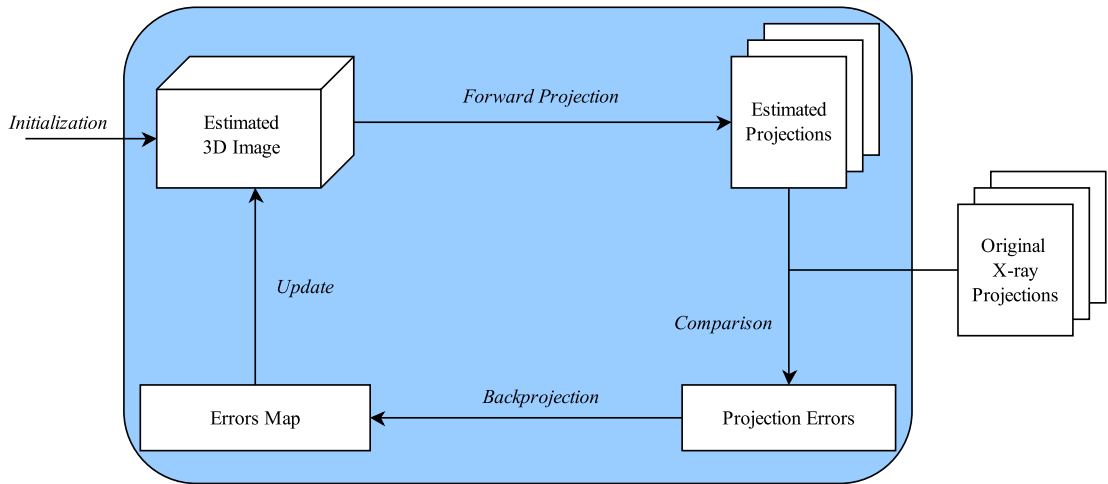


Figure 2.4: Schematic of iterative image reconstruction algorithm for DBT.

process for DBT. The method used for the calculation of A is independent of the algorithm used — the same A can be used in all iterative algorithms. To sum up, iterative algorithms in DBT can be seen as a step-by-step procedure to determine a 3D image \hat{x} as close as possible to the real attenuation coefficients that originated a certain set of X-ray projections, Y . In Figure 2.4 each loop represents one iteration. Each step of the diagram will be exposed in detail, for the relevant algorithms for this work, since it is essentially here where they differ.

In iterative algorithms each successive estimation should converge to the solution of the problem. However, due to the noise present in the "Original X-ray projections", as the iterative procedure proceeds, firstly the reconstruction converges to an image that can be recognized and then diverges to noise [94]. Therefore, in order to obtain the best results, it is important to know when to stop the iterations.

In this work, three different iterative algorithms will be implemented, one algebraic algorithm, SART, and two statistical algorithms, ML-EM and OS-EM.

2.3.1 Simultaneous Algebraic Reconstruction Technique

The Simultaneous Algebraic Reconstruction Technique (SART) is a particular instance of algebraic reconstruction methods. These methods pose the problem of reconstruction from projections as a set of simultaneous equations — Equation 2.2 to 2.5. Multiple algebraic algorithms have been proposed to solve this problem. The classical Algebraic Reconstruction Technique (ART) [103] has fast convergence speed but will converge to a least squares solution which can be very noisy for DBT. In order to improve ART, variations of its implementation have been proposed in which are included both the Simultaneous Iterative Reconstruction Technique (SIRT) [104] and the SART [105] methods. SIRT resulted in good noise suppression but had slow convergence rates and the reconstruction could be overly smoothed [49]. SART was the chosen algebraic method because it

combines the high convergence speed of ART with the noise suppression of SIRT, producing a superior reconstruction result. Moreover, ART is not suitable for the GPU, since each iteration only processes a single projection line.

For SART, an update is performed after all rays in one projection have been processed. The update step is given by:

$$\hat{x}_j^{k+1} = \hat{x}_j^k + \lambda \frac{\sum_{y_i \in Y_n} a_{ij} \left(\frac{y_i - \sum_j a_{ij} \hat{x}_j^k}{\sum_j a_{ij}} \right)}{\sum_{y_i \in Y_n} a_{ij}} \quad (2.7)$$

where λ is known as the relaxation parameter and its value is chosen in order to maximize Signal-to-Noise Ratio (SNR).

It is possible to relate Equation 2.7 to the scheme presented in Figure 2.4. By multiplying the estimate resulting from the previous iteration by the system matrix, we are calculating the "Estimated Projections" (*Forward Projection* process). Further, these "Estimated Projections" are compared with the "Original X-ray Projections" (*Comparison* process) by subtraction. The next step consists in multiplying the difference obtained by the equivalent system matrix element, performing the *Backprojection* operation. Finally, an *Update* of the estimate from the previous iteration is performed by adding the correction factor obtained from the *Backprojection* operation.

2.3.2 Maximum Likelihood – Expectation Maximization

In the Maximum Likelihood – Expectation Maximization (ML-EM) method it is assumed that the measurements follow Poisson statistics. The ML-EM formula for emission tomography [106] is given by:

$$\hat{x}_j^{k+1} = \frac{\hat{x}_j^k}{\sum_i a_{ij}} \sum_i a_{ij} \frac{Y_i}{\sum_t a_{it} \hat{x}_t^k} \quad (2.8)$$

In [107] an algorithm for transmission tomography is proposed. However, an exact analytical solution for this problem cannot be specified, being a mathematically more difficult problem than for emission tomography. Thus, and since the algorithm for emission tomography has become much more popular than the one for transmission [108], [109], we chose to focus our work on the emission tomography formulation.

Similarly to SART, Equation 2.8 of ML-EM can be related to the scheme presented in Figure 2.4 in the following way: *Forward Projection* — $\sum_t a_{it} \hat{x}_t^k$; *Comparison* — dividing Y_i by the result from the *Projection*; *Backprojection* — the results of the *Comparison* are summed and weighted by the system matrix elements; *Update* — multiplying the result of the *Backprojection* by the estimation obtained from the previous iteration.

The ML-EM algorithm has two major limitations when applied: convergence speed of the algorithm, which is very slow, and stability, since with noisy data the method

will reach a convergence point that corresponds to a noisy solution. The latter problem has two possible solutions: using a filter to the reconstructed image aiming to smooth the solution, or stopping the algorithm before it reaches the convergence but while the image is smooth.

2.3.3 Ordered Subsets – Expectation Maximization

In order to overcome the very slow convergence rate of ML-EM algorithm, it is possible to group together multiple projections and update the attenuation coefficients based on each one of the subsets. This is the principle underlying the Ordered Subsets – Expectation Maximization (OS-EM) algorithm [110].

The equation that describes OS-EM iterative process is the following:

$$\hat{x}_j^{k+1} = \frac{\hat{x}_j^k}{\sum_{i \in S_n} a_{ij}} \sum_{i \in S_n} a_{ij} \frac{Y_i}{\sum_t a_{it} \hat{x}_t^k} \quad (2.9)$$

being very similar to the Equation 2.8 of ML-EM with the difference that it is just applied to a specific subset S_n . Thus, each iteration of OS-EM is composed of n subiterations leading the estimated image to be updated n times per iteration. This enables a faster convergence speed but leads to the sooner appearance of significant noise.

2.4 Parallel Programming

Today, all computer architectures include multiple Central Processing Units (CPUs). In order to take advantage of such architectures new programming techniques are needed, sometimes associated with the scientific area of parallel programming. Parallel programming aims to mimic the natural world in the sense that many complex events happen at the same time. Thus, parallelism is a known and ubiquitous concept (e.g. in a house construction several workers can perform separate tasks simultaneously). In medical physics, many computing applications can be formulated as data-parallel tasks, which can reduce the processing times. The move from sequential programming to parallel programming has been a major change in the field of computer science.

Recently, the performance of a single-core processor has stopped obeying Moore's law [111] (i.e. stopped doubling the performance every 18 months) due to excessive power dissipation at GHz clock rates. Therefore, the need to make complex algorithms computationally feasible led to thinking about parallelism in computer programs and to an increase in the number of cores. One of the aspects of this hardware evolution has been the widespread use of what are called accelerators — devices that are normally installed in the system bus (PCI) and that perform intensive computations offloaded by the main CPU(s). Accelerators can be custom built for a specific purpose and thus dedicate silicon to special functions, making them particularly effective in the tasks they were designed for. The most successful use of accelerators in recent years has been the use of

GPUs for performing scientific calculations. Particularly, medical physics is increasingly converting its algorithms to parallel computing architectures to exploit the capabilities of the GPUs for practical processing times. Although there is some controversy about this, it is generally believed that GPUs offer a better relationship performance/cost when compared with conventional CPUs; the same applies to the relationship performance/-consumption. Naturally, some problems remain, the most important being the difficulties in programming GPUs, or more specifically the difficulty in achieving the performance levels of the raw hardware. The use of GPUs in irregular problems (those that do not conform to the data-parallel paradigm) is also an open question. Nevertheless, GPU speed grows much faster (2.4 times/year) than the Moore's law for CPU [112]. In fact, NVIDIA [113] indicated that from 2008 to 2014 the number of university courses on GPU computing increased from 60 to 738.

2.4.1 GPU

The GPU was originally designed for high-speed graphics, having emerged as a platform for running massively parallel computing. In fact, the video game industry had extreme importance in the fast-growing of GPUs [112], aiming to advance massive number of floating-point calculations. Thus, this hardware has the advantage of supporting floating-point arithmetic which is an asset for processing medical physics data. Nowadays, GPU stands as a reference in medical physics, each time with more on-going research such as in MRI [114], [115], ultrasound [116], [117] and CT [118], [119].

The reason of the large discrepancy in floating-point capability between the CPU and the GPU lies in the differences in the design of the two types of processors. CPU is optimized for sequential code performance, while GPU is specialized for compute-intensive, highly parallel, multithreaded, manycore processor. Therefore, many applications execute sequential parts on the CPUs — when it is not possible to parallelize these portions — and numerically intensive parts on the GPUs. In order to evaluate the maximum expected improvement to an overall system (*speedup*), the Amdahl's law [120] provides a theoretical upper limit on parallel speedup:

$$speedup = \frac{1}{r_s + \frac{r_p}{n}} \quad (2.10)$$

where n is the number of processors that run the same portion of code, and r_s and r_p the ratio of the sequential and parallelized portions, respectively. By taking into account that $r_s + r_p = 1$ and considering n to tend to infinity, Equation 2.10 can be reduced to:

$$speedup = \frac{1}{1 - r_p} \quad (2.11)$$

which shows that the higher the parallelization, the higher the *speedup* that can be achieved. Thus, in order to reduce execution time of a real application, one should take into account

that the time spent in the parallel portion should be the vast majority, reducing the component $1 - r_p$ to the smallest possible value. However, Amdahl's law assumes that there are no costs for CPU-GPU communications, which in reality have an important impact in the degradation of performance and must be taken into account.

Several Application Programming Interfaces (APIs) expose GPUs to the developer in a C-like programming paradigm. These commonly used APIs include, among others, NVIDIA CUDA and OpenCLTM software⁴ [122]. CUDA was the API chosen to be used in this dissertation, being currently the most popular GPU computing API, released in 2007 by NVIDIA [123]. Nevertheless, all the concepts are rapidly applied to other APIs. In fact, who are familiar with CUDA can pick up OpenCL relatively easy, due to the similarity between the key concepts and features. Therefore, it is fair to compare OpenCL with CUDA since both can be executed on NVIDIA GPUs. However, their goals are rather different, CUDA is specific to NVIDIA GPUs (directly connected to the execution platform) whereas OpenCL promises the key feature of portability (through its abstracted memory and execution model). Moreover, being CUDA more mature than OpenCL, it has been used and optimized more heavily. CUDA provides the best performance [124].

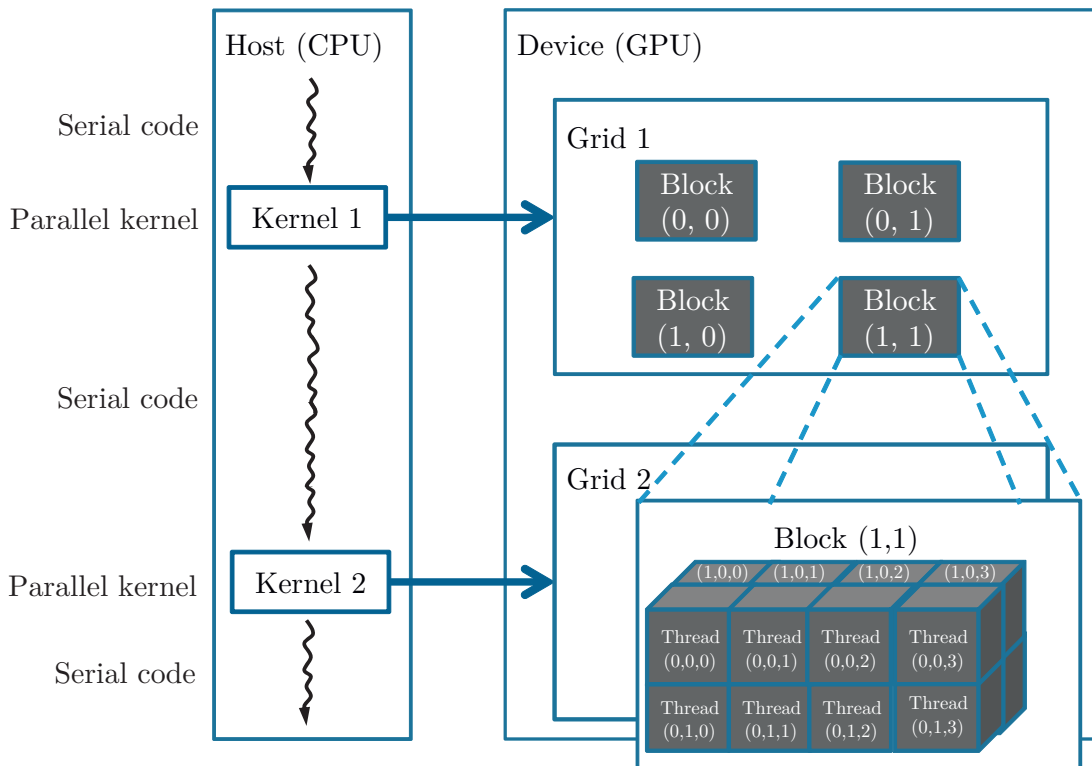


Figure 2.5: Execution of a CUDA program, using a multidimensional example of a CUDA grid organization. Adapted from [121].

⁴OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

2.4.2 CUDA

Compute Unified Device Architecture (CUDA) is a parallel computing architecture developed by NVIDIA which allows the programmer to use the C programming language (besides the existence of other supported languages such as FORTRAN). CUDA provides, at its core, three key abstractions — a hierarchy of thread groups, shared memories, and barrier synchronization — that are simply exposed to the programmer as a set of language extensions. Furthermore, CUDA has already reached scientists throughout industry and academia [125].

The structure of a CUDA program reflects the coexistence in the computer of a host (CPU), which executes the sequential code, and one or more devices (GPUs), responsible for the execution of the parallelization (Figure 2.5). The host launches the kernel, a multithreaded program, in which the parallel task is executed. Kernels are like C functions that instead of being executed once they are executed N times in parallel by N different threads, which are grouped into a grid of blocks (thread hierarchy). There is a limit number of threads per block and the thread blocks are required to execute independently (must be possible to execute in an arbitrary order), which allows to boost the parallel factor of the GPUs.

The CUDA memory model mimics the hierarchy of the threads, it coordinates the memory access per-thread, per-block and per-grid, dividing the memory between grids, and structuring the memory spaces (Figure 2.6). In this memory scheme, the global and constant memory can be written and read by the host (i.e. the host has full access). Global

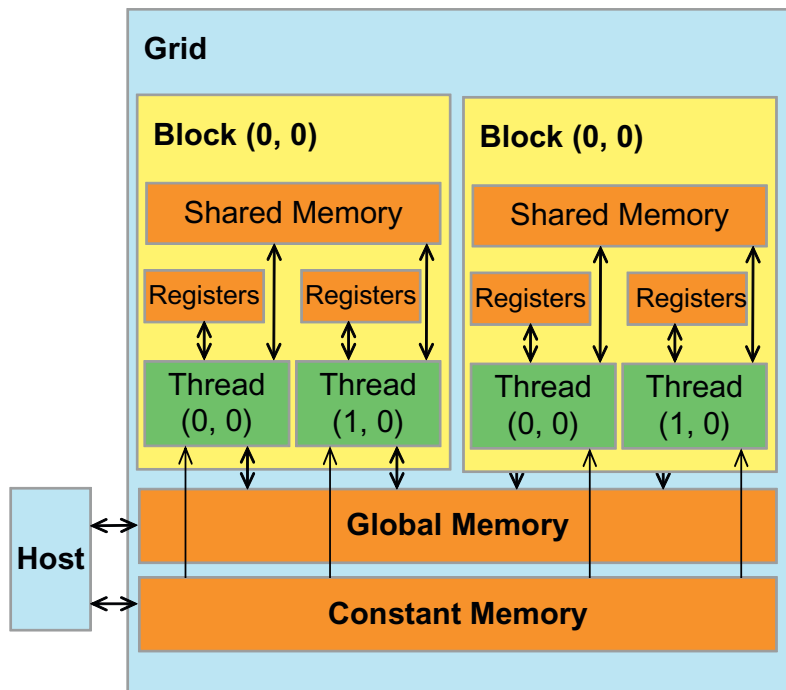


Figure 2.6: CUDA memory model. Adapted from [121].

memory is randomly accessible for reading and writing by all threads in the application, having the potential of traffic congestion (which prevents all but a few threads from making progress). Shared memory is an efficient means for threads to cooperate by sharing data and synchronizing their execution, coordinating memory accesses. Registers and shared memory are located on the GPU chip and can be accessed at very high speed in a highly parallel manner. However, global memory can store far more data and this data persists beyond the lifetime of the kernels. Local memory is allocated to threads for storing their private data. Each thread in this memory model can:

- Read/write per-thread registers.
- Read/write per-thread local memory.
- Read/write per-block shared memory.
- Read/write per-grid global memory.
- Read only per-grid constant memory.

In parallel programming, often all threads need to do their computation before processing the next phase of their execution. This is known as barrier synchronization (Figure 2.7) — point in the program where all threads stop and wait; only when all threads have reached the barrier, they can proceed. However, this ability to synchronize imposes execution constraints. Furthermore, CUDA does not allow barrier synchronization for threads in different blocks. In fact, this explains CUDA being able to execute blocks in any order — none of them have to wait for each other.

In CUDA, conflicts can arise when the same memory location is attempted to be written simultaneously by multiple threads. In such a case, only one thread succeeds in

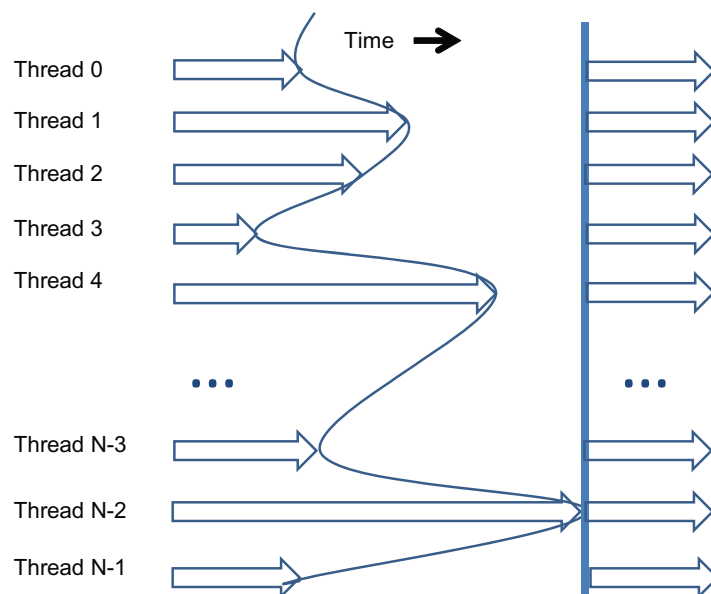


Figure 2.7: An example of barrier synchronization [121].

the writing. In order to overcome this major issue, threads must use an atomic operation (operation that cannot be interrupted by concurrent operations). The GPU processes conflicting atomic writes in a serial manner, avoiding data write problems.

An optimal performance with CUDA can only be achieved with a careful allocation of compute and storage resources. A first priority should be to parallelize as much sequential code as possible. In order to minimize global memory access, GPU codes should use shared memory (e.g. by grouping threads into blocks such that threads work on common data). Constant memory can provide a faster access than global memory when memory is only read. However, for many applications it is not possible to avoid the use of global memory, being necessary to carefully coordinate memory accesses. Additionally, host-device memory transfer should be minimized for high-performance computing.

One concern with parallel programming is the reliability of memory operations. Particularly, in medical physics, image reconstruction can be tampered due to errors introduced while reading or writing memory and therefore should be dealt with caution.

2.4.3 State of the Art of Image Reconstruction using Parallel Programming

Several studies have already demonstrated the efficacy of the GPU implementation of reconstruction algorithms. Iterative algorithms are computationally challenging, making GPU acceleration a need. Analytical algorithms are fully parallel whereas the iterative ones are essentially sequential. Thus, methods performing none or almost none computation within each iteration should not be implemented on the GPUs — insufficient parallel workload. This is the reason why, for instance, ART is not suitable for GPU (each iteration only processes a single projection line) but SART is. Besides SART, ML-EM and OS-EM, several other iterative algorithms were also adapted to the GPU, including the ordered-subsets convex algorithm [126], [127], and Total Variation (TV) reconstruction [128], [129]. In [130] the authors show that their implementation of both analytical and iterative reconstructions in GPU for CT allow speed-ups of more than one order of magnitude and the quality of the image is comparable. Further, the state of the art of the relevant algorithms for this work is provided:

- **SART.** Keck *et al.* [131] implemented SART on the GPU using CUDA, obtaining a higher speed-up than 64.
- **ML-EM.** ML-EM was already implemented for DBT by Goddard *et al.* [132] on a GPU, in which they reported a 113-fold speed-up in processing time and verified that the high image quality was maintained. In [133], Schaa *et al.* accelerated ML-EM on multiple CUDA-enabled GPUs for DBT, reducing the execution time for eight iterations to less than 20 seconds.
- **OS-EM.** Pratz *et al.* [134] implemented OS-EM with a GPU approach reconstructing 51 times faster than an equivalent CPU implementation and with no differences in the image quality.



Materials and Methods

The present chapter presents an overview of the main technical characteristics and the followed methodology in this dissertation. Firstly, the DBT system that was used is described in Section 3.1. Then, in Section 3.2 an IDL^{®5} implementation already developed within the group, which is the basis of this work, is exposed, particularly the system matrix calculation that motivates the use of GPUs. In order to familiarize the reader with CUDA programming concepts, Section 3.3 addresses this matter and provides the advantages of using GPU-accelerated libraries and development tools. Finally, a new DBT-based image reconstruction method is explained in Section 3.4.

3.1 DBT System

The DBT system used in this work was the Siemens MAMMOMAT Inspiration (Figure 3.1) installed in Hospital da Luz (Departamento de Imagiologia), Lisbon, Portugal.

The Siemens MAMMOMAT Inspiration system enables both the acquisition of DBT and DM. In DBT, breast positioning and system operation do not change comparing to the DM system, which enables radiologists to easily adapt to the system. For the DBT acquisition, the X-ray tube arcs over the breast with an angular range of 50° (-25° to +25°), the widest angle in the industry. It performs pulsed acquisition in the continuous sweep, avoiding mechanical instabilities when compared to the step-and-shot method, and obtains 25 projections — approximately 2° angle increment per image — on a stationary detector, each projection sizing 19.7 MB [136]. The radiation dose is equivalent to two incidences of mammography. Currently, the DBT acquisition takes approximately 20 s [137]. From the 25 projections it is then possible to reconstruct images 1 mm thick.

⁵IDL is a registered trademark of Exelis Visual Information Solutions.



Figure 3.1: Siemens MAMMOMAT Inspiration [135].

The analytical reconstruction algorithms that are used in this system — FBP — take approximately 60 s to reconstruct the 3D image, with a total file size of 2.1 GB for 60 mm thickness [136].

The exact geometry of the DBT system while performing the exam is essential for the reconstruction — Figure 3.2. Thus, the exact angle of each projection is measured on-line during the scan to be used in the reconstruction algorithm. The rotation center is 4.7 cm above the detector surface and the distance between the X-ray tube and the axis of rotation is 62.5 cm. The detector is an amorphous selenium flat panel with an array of 2816×3584 bins in which each bin has $85 \mu\text{m} \times 85 \mu\text{m}$, rendering an active area of approximately $24 \text{ cm} \times 30 \text{ cm}$.

This work uses the projections of a DBT examination of a 60 mm compressed breast. The 3D reconstructed image is intended to be 1 mm thick, thus the number of slices, N_{slices} , is 60. The Field Of View (FOV) is defined as the volume irradiated by the X-ray source and its dimension is given by:

$$Detector_{Dim_x} \times Detector_{Dim_y} \times N_{\text{slices}} = 2816 \times 3584 \times 60 \quad (3.1)$$

resulting in 605,552,640 voxels. Due to memory constraints, this is not the amount of voxels used in this work. This matter will be addressed in Section 3.4.

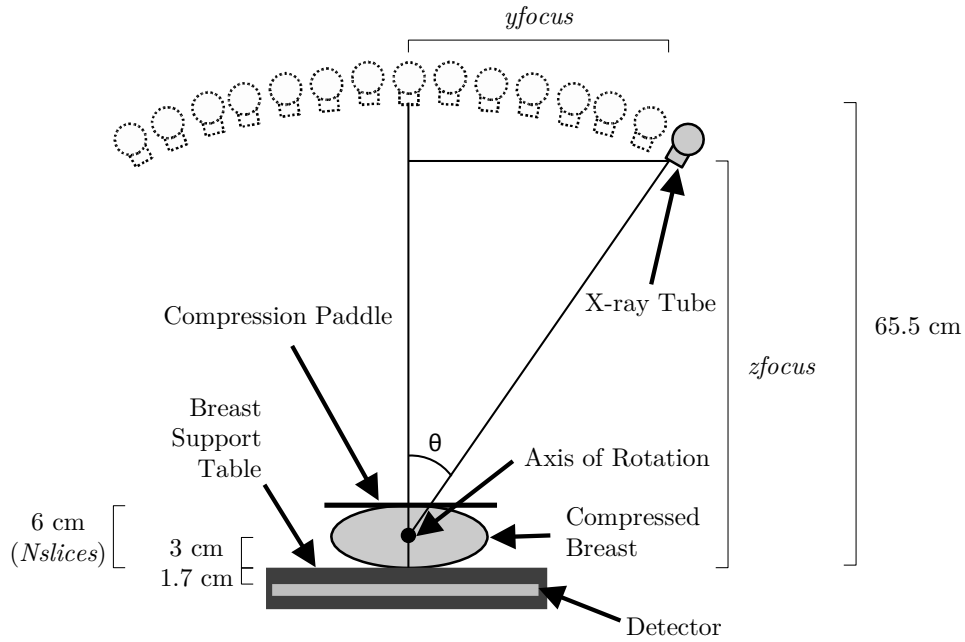


Figure 3.2: Diagram of the Siemens MAMMOMAT Inspiration system performing DBT. An X-ray source covers a 50° arc (-25° to $+25^\circ$, θ) emitting low-dose X-rays at 25 locations (in this image is only presented 15 locations for simplicity). Twenty five corresponding projections are acquired by the detector. The most relevant dimensions of the system are: 65.5 cm between the X-ray tube in the initial state (when $\theta = 0^\circ$) and the breast support table, 3 cm between the axis of rotation and the breast support table and 1.7 cm between the breast support table and the detector. The compressed breast in this work is 6 cm and it is related with the number of slices in the 3D reconstructed image (N_{slices}). y_{focus} and z_{focus} represent, respectively, the coordinates y and z of the detector, both depending on θ . Adapted from [133].

3.2 IDL Implementation

An implementation for the DBT image reconstruction made in IDL is the basis of this work. IDL stands for Interactive Data Language and is a vectorized programming language used for data analysis, in which it is possible to transform complex numerical data into meaningful visualizations. IDL provides support for several operating systems, including Microsoft® Windows®⁶, Mac® OS X®⁷ and Linux®⁸. IDL is commonly used in the field of medical imaging [138]–[142] as it is a simpler and therefore easier programming language to be used by non-computer scientists when compared with, for example, CUDA and C which are both lower-level programming languages. IDL development requires the purchase of a license thus, the option to use GNU Data Language (GDL), a free alternative, was also taken into consideration but discarded due to the fact that it does

⁶Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and other countries.

⁷Mac and OS X are registered trademarks of Apple Inc., registered in the United States and other countries.

⁸Linux is a registered trademark of Linus Torvalds in the United States and other countries.

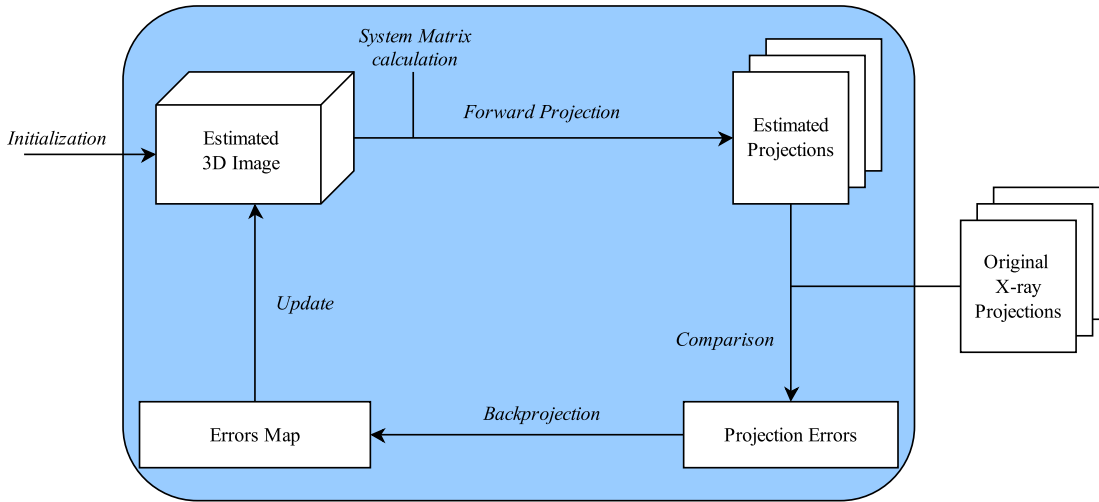


Figure 3.3: Schematic of the iterative image reconstruction algorithms for DBT applied to the IDL implementation.

not have yet every needed functionality.

The already existent IDL implementation in the research group provides a DBT reconstruction with several algorithms: SART, ML-EM and OS-EM. The TV algorithm is also available in the IDL implementation which can be used in combination with each one of the other three as post-processing to minimize the noise. Every iterative algorithm in this implementation is according to the scheme presented in Figure 3.3, in which the system matrix is calculated on the fly due to memory constraints.

IDL's high-level language prevents programmers to know the details of the internal structures of a computer which can be an issue when one thinks about accelerating computational processes. IDL lets programmers avoid several difficult elements of low-level languages (e.g. memory management), but has the drawback to tend to run slower. In fact, compiled languages like C and CUDA have several advantages over interactive languages, as compilers can often organize the object code so that it is optimized to execute very quickly. In this work, a simple IDL application is not a viable solution since the implementation of the DBT reconstruction requires optimisations that only lower-level languages allow. Multi-threading capability is available in IDL [143] to accelerate specific numerical computations on machines with multiple processors (including binary and unary operators, array manipulation and type conversion routines), which was tested in this work for the already existent IDL implementation. However, when top-notch efficiency is required, developers are turning to the use of GPUs (despite its difficulty).

As seen in Sub-section 2.4.1, two different APIs were considered (CUDA and OpenCL). With IDL, there is other possibility called GPULib^{®9} [144] which enables to access high performance computing with minimal modifications to the existing IDL programs and without required knowledge about GPU programming. This possibility was disregarded

⁹GPULib is a registered trademark of Tech-X Corporation.

for two different reasons: cost and maladaptation between the GPULib computing model and the algorithm used for the system matrix calculation. This led us to choose the most popular option: CUDA.

Multiple parts of the DBT iterative reconstruction process (Figure 3.3) can be considered when thinking about parallelization, mainly:

1. The system matrix calculation.
2. The forward projection.
3. The backprojection.

In this work the calculation of the system matrix, detailed in the next Sub-section, is parallelized using CUDA as it is the most computationally intensive part.

3.2.1 System Matrix Calculation

As noted in Section 2.3, the system matrix describes the geometric model of the transmission and detection processes being the very core of all iterative algorithms (regardless if algebraic or statistical). The system matrix calculation is independent of the type of iterative algorithm and depends only on the geometry of the DBT system.

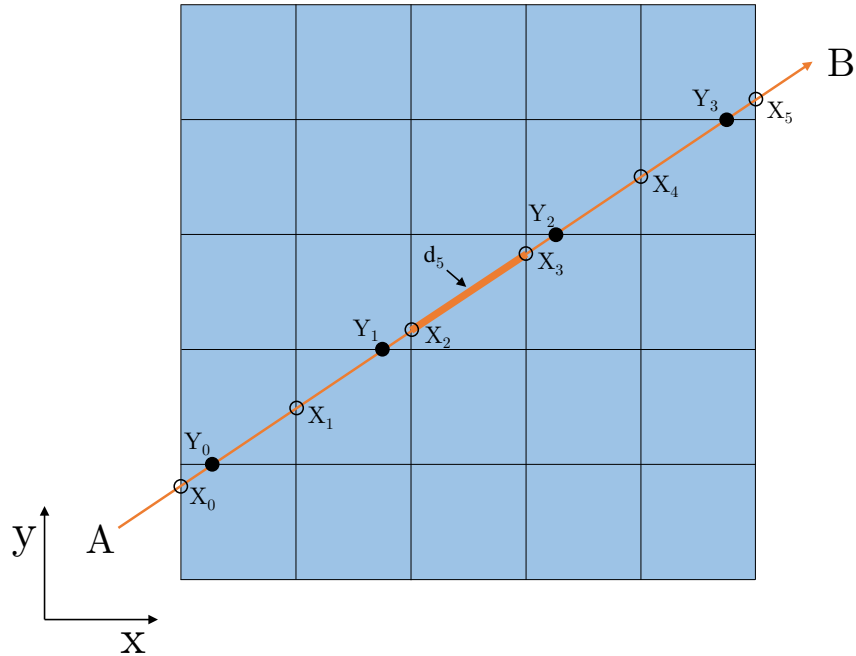


Figure 3.4: Two-dimensional example on the xy plane of the ray driven approach. The \overline{AB} ray intersects the horizontal (filled circles) and vertical (open circles) lines. d_5 represents the 5th intersection length of the ray within the image — Euclidean distance between the points with x -coordinate X_2 and X_3 . The extension to three-dimensional is straightforward.

The calculation of the system matrix in the IDL implementation uses a ray driven approach [145], [146]. This approach efficiently calculates the distances between intersections by following the path of each ray through the volume. Without loss of generality, Figure 3.4 illustrates a two-dimensional example (xy plane) of a single ray path with its intersections with the vertical and horizontal lines. An example of intersection length is shown by d_5 , the 5th intersection length of the ray within the image, which is calculated through the Euclidean distance between the points with x -coordinate X_2 and X_3 . The generalization to three-dimensional is straightforward as the ray from point A (A_x, A_y, A_z) to point B (B_x, B_y, B_z) may be represented parametrically as

$$\begin{aligned} X(\alpha) &= A_x + \alpha(B_x - A_x) \\ Y(\alpha) &= A_y + \alpha(B_y - A_y) \\ Z(\alpha) &= A_z + \alpha(B_z - A_z) \end{aligned} \quad (3.2)$$

where $0 \leq \alpha \leq 1$, being zero at point A and unity at point B .

The ray driven approach applied to the DBT system consists on drawing rays from each detector's bin i to the X-ray beam focus, determining the intersections of each ray R_i with the FOV (Figure 3.5). The intersection lengths are then calculated, which after normalization represent the relative contribution of the voxels to the attenuation of the corresponding ray. This means that, for a specific R_i :

$$\sum_j a_{ij} = 1 \quad (3.3)$$

where a_{ij} is a system matrix element which gives the influence of the j^{th} voxel in the

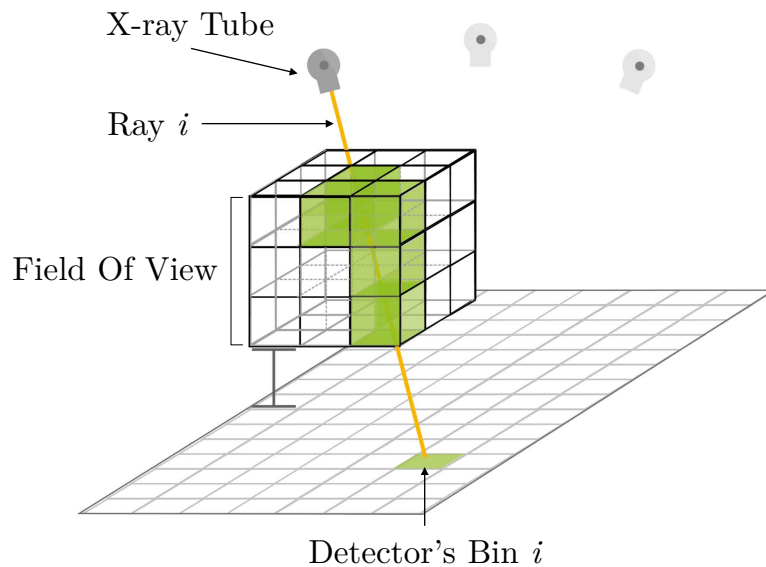


Figure 3.5: Illustration of a ray i (R_i) from a detector bin i to the X-ray beam focus, illustrating that R_i is attenuated only by voxels that are crossed by it (green voxels).

i^{th} ray. For each projection, a different system matrix is required so that it is possible to reproject or update with the correct model.

In order to compute the intersections, it is necessary to solve Equation 3.2. For this process several specific geometric considerations of the system must be considered, mainly the position of each bin of the detector, point A , and the position of the X-ray source focus, point B . A is calculated as follows:

$$\begin{aligned} A_x &= (n_x + 0.5) \times xbinsize \\ A_y &= (n_y + 0.5) \times ybinsize \\ A_z &= -17 \text{ mm} \end{aligned} \quad (3.4)$$

in which $xbinsize = ybinsize = 85 \mu\text{m}$ and $n_x, n_y \in \mathbb{N}$ such that $0 \leq n_x < DetectorDim_x$ and $0 \leq n_y < DetectorDim_y$. Regarding to the calculation of B :

$$\begin{aligned} B_x &= \frac{DetectorDim_x}{2} \times xbinsize = \frac{2816}{2} \times 0.085 \text{ mm} \\ &= 119.68 \text{ mm} \\ B_y &= d_1 \times \sin(\theta) + \frac{DetectorDim_y}{2} \times ybinsize \\ &= 625 \text{ mm} \times \sin(\theta) + \frac{3584}{2} \times 0.085 \text{ mm} \\ &= 625 \text{ mm} \times \sin(\theta) + 152.32 \text{ mm} \\ B_z &= d_1 \times \cos(\theta) + d_2 \\ &= 625 \text{ mm} \times \cos(\theta) + 30 \text{ mm} \end{aligned} \quad (3.5)$$

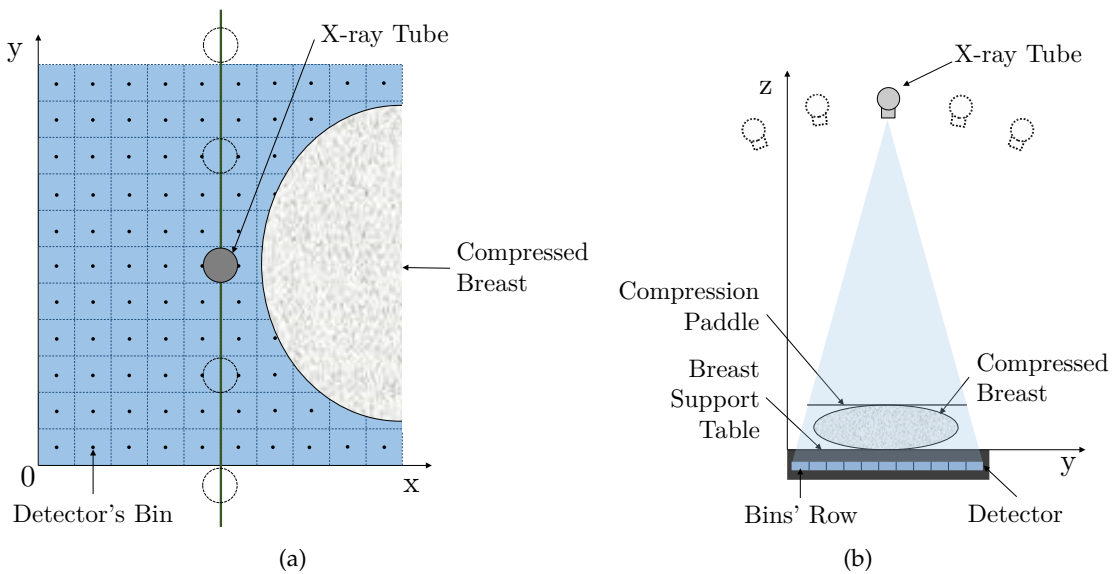


Figure 3.6: Focus positions on (a) xy and (b) yz planes.

where d_1 is the distance between the axis of rotation and the X-ray tube and d_2 is the distance between the breast support table and the axis of rotation. B_x is constant throughout the DBT acquisition which implies geometrical symmetry (Figure 3.6). Thus, there is no need to calculate the system matrix on the right-half of the x -dimension. Therefore, every ray drawn from the bins with coordinates $(B_x \pm x_{bin}, y_{bin}, 0)$ are processed equally.

With the calculation of A and B it is now possible to find the intersections of the ray with the grid of voxels. Firstly, we assume that R_i always intersects a given axis (x , y or z) when a point of the ray is equal to a natural number times the bin size along that direction. Secondly, for each intersection coordinate in an axis, the parameter α is calculated which allows to determine the coordinates of the remaining two axis. For instance, from Equation 3.2 for the intersections with the x -coordinate:

$$\begin{aligned}\alpha &= \frac{X(\alpha) - A_x}{B_x - A_x} \\ Y(\alpha) &= A_y + \alpha(B_y - A_y) \\ Z(\alpha) &= A_z + \alpha(B_z - A_z)\end{aligned}\tag{3.6}$$

being α first calculated, followed by $Y(\alpha)$ and $Z(\alpha)$. Thirdly, the FOV dimensions are considered in order to remove extra intersections that occur outside of the FOV — the breast thickness to be reconstructed is much smaller than the detector to X-ray tube distance, which results in obtaining many ray intersections in planes above the reconstruction volume. Finally, the calculation of the Euclidean distances between consecutive intersections is performed (N intersections lead to $N - 1$ distances) and each distance is attributed to its specific voxel. In order to identify the voxel coordinates to which a distance is associated, each ray is considered to be in the direction from the bin to the X-ray tube and the coordinates from the exit intersection of the voxel is considered. The ray

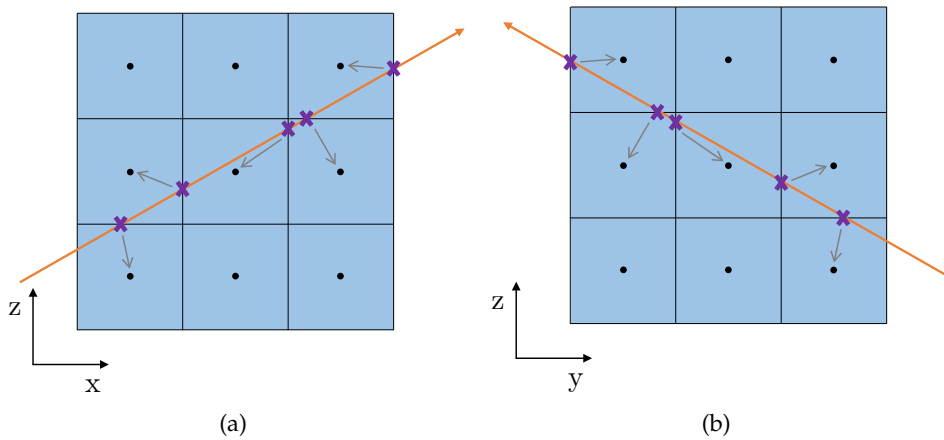


Figure 3.7: Matching of the intersection points of a ray and the voxels on the (a) xz plane (which has always a positive slope) and (b) yz plane (which in this case has a negative slope).

slope can be either positive or negative and the voxel coordinates are calculated differently. If:

- positive — voxel coordinates are obtained by subtracting 1 from intersection coordinates anytime the latter is multiple of the bin size and by rounding otherwise (Figure 3.7 (a));
- negative — all intersection coordinates are rounded (Figure 3.7 (b)).

Based on the fact that there is only need to consider the rays on the right-half of the x -dimension, all the considered rays have a positive slope in xz plane. In the yz plane the ray slope can be both positive and negative.

3.3 Fundamentals of CUDA Programming

The system matrix calculation can be parallelized by distributing the rays between available threads. Therefore, basic concepts regarding CUDA programming are presented in this section so the reader can better understand the proposed method to reconstruct DBT images (Section 3.4).

The main concepts behind CUDA programming are introduced by outlining how they are exposed in C. As presented in Sub-section 2.4.2, a kernel is a C function that, when called, is executed N times in parallel by N different CUDA threads, as opposed to only once like regular C functions. A kernel function (`kernel_name`) runs on the device and is defined using the `__global__` declaration as:

```
__global__ void kernel_name(arguments_declaration)
```

To invoke a kernel one must specify, besides the arguments as done in a C-function, the number of thread blocks (`number_blocks`) and the number of threads per block (`number_threads`) to be executed:

```
kernel_name<<<number_blocks, number_threads>>>(arguments);
```

in which both variables `number_blocks` and `number_threads` can be of type `int` or `dim3`. The total number of threads is equal to: `number_blocks × number_threads`. Each thread that executes the kernel has a unique combination of thread ID and block ID that is accessible, within the kernel, through the built-in variables `threadIdx` and `blockID`, respectively. Both these variables are 3-component vectors, so that each one can be identified using a one-dimensional, two-dimensional, or three-dimensional index. The dimension of threads in a block is accessible within the kernel through the built-in `blockDim` variable and the dimension of blocks in a grid by `gridDim`, which are both also 3-component vectors.

CUDA memory constraints must be taken into consideration while programming to enhance the execution times. A relevant constraint addressed in this work is the limited

number of threads per block — current GPUs may contain up to 1024 threads in a thread block. The reason for this is that every thread in a block resides on the same processor core to cooperate between each other by making use of that core’s limited memory resources. However, for threads within a block to share data they must coordinate their memory access by synchronization — one can specify synchronization points in the kernel with the function `__syncthreads()`.

As seen in Table 3.1, CUDA uses the C-functions to allocate memory and free it on the host, but in the device instead of using `malloc()` and `free()`, uses `cudaMalloc()` and `cudaFree()`, respectively. While `malloc()` returns a pointer to the allocated object, `cudaMalloc()` writes to the pointer variable of which the address is given as the first parameter, having two parameters and a return value to report errors. In order to transfer data from host memory to device memory and vice-versa, `cudaMemcpy()` is typically used — similar to the C-function `memcpy` but with the extra argument of the type of transfer (usually `cudaMemcpyHostToDevice` or `cudaMemcpyDeviceToHost`), which specifies the direction of the copy.

Table 3.1: Functions used on the host and device to allocate and free memory in CUDA.

	Host (CPU)	Device (GPU)
Allocate memory	<code>malloc()</code>	<code>cudaMalloc()</code>
Free memory	<code>free()</code>	<code>cudaFree()</code>

For a better understanding of the major differences between C and CUDA programming, a simple problem of vector addition is solved using these two languages (Listings 3.1 and 3.2). The problem is the following: given A and B, two float vectors of dimension n , perform one pair-wise addition and save it on C, another float vector.

Listing 3.1: Vector Addition — C Code.

```

1 void vecAdd(float* A, float* B, float* C, int n) {
2     int i;
3     for (i = 0; i < n; i++)
4         C[i] = A[i] + B[i]
5 }
6
7 int main() {
8     int n = ...; // size of the vectors
9     size_t size = n * sizeof(float);
10
11     // Allocate input vectors h_A and h_B and output vector h_C
12     float* h_A = (float*)malloc(size);
13     float* h_B = (float*)malloc(size);
14     float* h_C = (float*)malloc(size);
15
16     // Initialize input vectors
17     ...
18

```

```

19  vecAdd(h_A, h_B, h_C, n);
20
21  // Free host memory
22  free(h_A);
23  free(h_B);
24  free(h_C);
25  }

```

Listing 3.2: Vector Addition — CUDA Code.

```

1  // Device code
2  // Each thread performs one pair-wise addition
3  __global__ void vecAdd(float* A, float* B, float* C, int n) {
4      int i = blockDim.x * blockIdx.x + threadIdx.x;
5      if (i < n)
6          C[i] = A[i] + B[i];
7  }
8
9  // Host code
10 int main() {
11     int n = ...; // size of the vectors
12     size_t size = n * sizeof(float);
13
14     // Allocate input vectors h_A and h_B and output vector h_C in host memory
15     float* h_A = (float*)malloc(size);
16     float* h_B = (float*)malloc(size);
17     float* h_C = (float*)malloc(size);
18
19     // Initialize input vectors (in the host)
20     ...
21
22     // Allocate vectors in device memory
23     float* d_A;
24     cudaMalloc(&d_A, size);
25     float* d_B;
26     cudaMalloc(&d_B, size);
27     float* d_C;
28     cudaMalloc(&d_C, size);
29
30     // Copy vectors from host memory to device memory
31     cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
32     cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
33
34     // Invoke kernel
35     // This makes sure that there are enough threads to cover all elements
36     // Run ceil(n/256.0) blocks of 256 threads each
37     dim3 DimGrid(ceil(n/256.0), 1, 1);
38     dim3 DimBlock(256, 1, 1);
39     vecAdd<<<DimGrid, DimBlock>>>>(d_A, d_B, d_C, n);
40
41     // Copy result from device memory to host memory

```

```
42  cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
43
44  // Free device memory
45  cudaFree(d_A);
46  cudaFree(d_B);
47  cudaFree(d_C);
48
49  // Free host memory
50  free(h_A);
51  free(h_B);
52  free(h_C);
53 }
```

These two different solutions for this simple problem illustrate several differences between C and CUDA programming. From these differences one can reason how parallel programming with CUDA can easily become a very hard tool to solve a complex problem. To help with this task, several GPU-accelerated libraries and development tools are included in the NVIDIA CUDA Toolkit.

3.3.1 GPU-Accelerated Libraries

The CUDA Toolkit includes the following GPU-accelerated libraries:

- **cuFFT** — Fourier transforms;
- **cuBLAS** — Complete BLAS (Basic Linear Algebra Subroutines) library;
- **cuSPARSE** — Sparse matrix library (a sparse matrix is a matrix in which most elements are zero);
- **cuRAND** — Random number generator;
- **NPP** — Primitives for image & signal processing;
- **Thrust** — Templated parallel algorithms & data structures;
- **CUDA Math Library** — High performance math routines.

Several other GPU-accelerated libraries are available online and can be easily incorporated in a CUDA program by just calling the library function. For an extensive list of libraries available today refer to [147].

The most relevant libraries that one can think of for the DBT image reconstruction problem are: cuBLAS, cuSPARSE, NPP and Thrust. NPP is discarded for the system matrix calculation besides being a possible candidate to be used in processing the reconstructed images. cuSPARSE is also rejected since in the IDL implementation of the system matrix calculation the resulting matrices' elements are mostly non-zero. Both cuBLAS and Thrust were the most adequate. However, Thrust presents more appropriate functions (which will be presented next) that were essential for this work, being therefore the selected CUDA library.

3.3.1.1 Thrust

Thrust [121] is a C++ template library for CUDA based on the Standard Template Library (STL) [148]. GPU-accelerated sort, scan, transform and reduction operations are some of the functionalities that can be used through this library. In order to use Thrust in a CUDA program it is necessary to have some concepts first, such as containers and iterators. Thrust provides two containers, `host_vector` (stored in host memory) and `device_vector` (stored in device memory), and each container has a pair of iterators, `begin()` and `end()`. For instance, having:

```
thrust::device_vector<int> d_vec(4);
```

it is possible to access the iterator `begin()` by:

```
d_vec.begin();
```

which returns the iterator at first element of `d_vec`. The `end()` would return an iterator one past the last element. For the mentioned example, the pair of iterators defines a sequence of 4 elements, acting like pointers that provide means to access the data stored in the containers. Several other specific iterators are used in Thrust, being one of the most relevant for this work the `zip_iterator` which takes multiple input sequences and yields a sequence of tuples.

The implementation of several algorithms in parallel are much harder comparing to the sequential implementation, such as sorting, stream compaction (also known as stream reduction) and a simple sum of every element in an array, which can be seen in parallel programming as a reduction operation. Thrust is a library of generic functions (independent of any particular type) that provides many of the standard algorithms:

- **Transformations** — apply an operation to each element in a set of input ranges;
- **Reductions** — use a binary operation to reduce an input sequence to a single value (e.g. a sum of an array of numbers is obtained by reducing the array through a plus operation);
- **Prefix Sums** — also known as scan operations or cumulative sum of a sequence of numbers;
- **Reordering** — supports for partitioning and stream compaction through several algorithms including `copy_if`, `remove_if` and `unique`;
- **Sorting** — sorts data or rearranges data according to a given criterion.

All these algorithms from Thrust have implementations for both the host and the device. Moreover, these functions are optimized in such a way that the programmer does not have to define the number of thread blocks and threads per block to be used in each function.

The system matrix calculation benefits from most of the Thrust algorithms. Particularly, reordering algorithms, which were necessary in this work to remove the intersections out of the FOV and the zero elements through `remove_copy_if`, and sorting algorithms, which were indispensable to sort the bins with the intersections before computing the distances through `sort_by_key`. For simplicity reasons, two brief examples concerning the `remove_copy_if` (Listing 3.3) and `sort_by_key` (Listing 3.4) functions are presented next:

Listing 3.3: `remove_copy_if` — Thrust function example.

```

1 const int n = 4;
2 int v[n] = {-2, 0, 1, 4};
3 int s[n] = { 0, 1, 0, 1};
4 int result[2];
5 thrust::remove_copy_if(v, v + n, s,
6     result, thrust::identity<int>());
7 // v remains {-2, 0, 1, 4}
8 // result is now {-2, 1}

```

Listing 3.4: `sort_by_key` — Thrust function example.

```

1 const int n = 4;
2 int keys[n] = { 1 , 4 , 2 , 7 };
3 char values[n] = {'a','b','c','f'};
4 thrust::sort_by_key(keys, keys + n,
5     values, thrust::greater<int>());
6 // keys is now { 7 , 4 , 2 , 1 }
7 // values is now {'f','b','c','a'}

```

The applications of these functions for the system matrix calculation are much more challenging since it is necessary to make use of more complex iterators, such as the type `zip_iterator`, and to create new specific operators which are more complex than the predefined thrust operators `identity<T>` and `greater<T>`, as seen in the above examples. The operators that are necessary for this work have to be manually programmed in order to compute exactly what is expected from the system matrix calculation.

3.3.2 Development Tools

The CUDA Toolkit includes the following development tools: Nsight integrated development environment, Visual Profiler, CUDA-GDB command line debugger and CUDA-MEMCHECK memory analyzer.

Nsight has two different editions: Visual Studio Edition and Eclipse Edition. The latter is part of the CUDA Toolkit Installer for Linux and Mac and was the tool explored in this dissertation due to the use of computers with Linux. The main functionalities that were exploited were the:

- **debugger**

- debugs code running on the host and on the device simultaneously;
- visualizes the variables of a program across multiple threads;
- maps the kernels and examines the state of the execution;
- sets breakpoints and executes single-step instructions;
- detects memory access errors as it includes CUDA-MEMCHECK.

- **profiler**

- accurately locates optimization opportunities;
- pinpoints potential performance problems within kernels.

A GPU that is running on Linux X Window SystemTM (X11TM)¹⁰, a windowing system for bitmap displays, cannot be used to debug a CUDA application, which means that the GPU must be exclusively used for debugging purposes. As the computers in this work contain a single GPU, the proposed solution is to remotely access the system through Secure Shell (SSH), a protocol for securely getting access to a remote computer. Another way to debug is by printing computed data through the bash command line. In this case, since it is not possible to print data directly from the GPU, it is always necessary to transfer it back to the host and then print it to the command line (as it is done with a C program). In this method it is also possible to use CUDA-MEMCHECK simply by passing the application's name as a parameter to CUDA-MEMCHECK on the command line, as shown next:

```
cuda_memcheck ./application_name.out
```

A GPU that is running X11 can still be used for profiling GPU applications. In fact, the Nsight Profiler is an excellent tool to use after the program is running and producing the expected results, delivering vital feedback for optimizing the application.

3.4 CUDA Integration in IDL

This dissertation proposes the development of a fast DBT-based image reconstruction method that includes the best attributes of the CPU and the GPU, where the latter is used in the most computationally expensive parts. Thus, a heterogeneous CPU+GPU configuration is used where each part of the computation is allocated to the most suitable processor. In 2012, Sharma *et al.* [149] developed hybridMANTIS, a Monte Carlo package, using a similar approach. They used a CPU-GPU technique with C+CUDA and were able to successfully reduce in hours the program execution, comparing to a CPU-implementation, by running parts in parallel.

Integration of CUDA in IDL (Figure 3.8) allows an incremental approach, replacing sequential parts of the computation by parallelized ones, as soon as available. This dissertation focuses in integrating the system matrix calculation completed on the GPU in the IDL implementation. The greatest advantage of this method is that the quality of results can be assessed by comparison with pure-IDL implementation. To the best of the author's knowledge this area of work has never been attempted before.

¹⁰X Window System and X11 are trademarks of The X.Org Foundation.

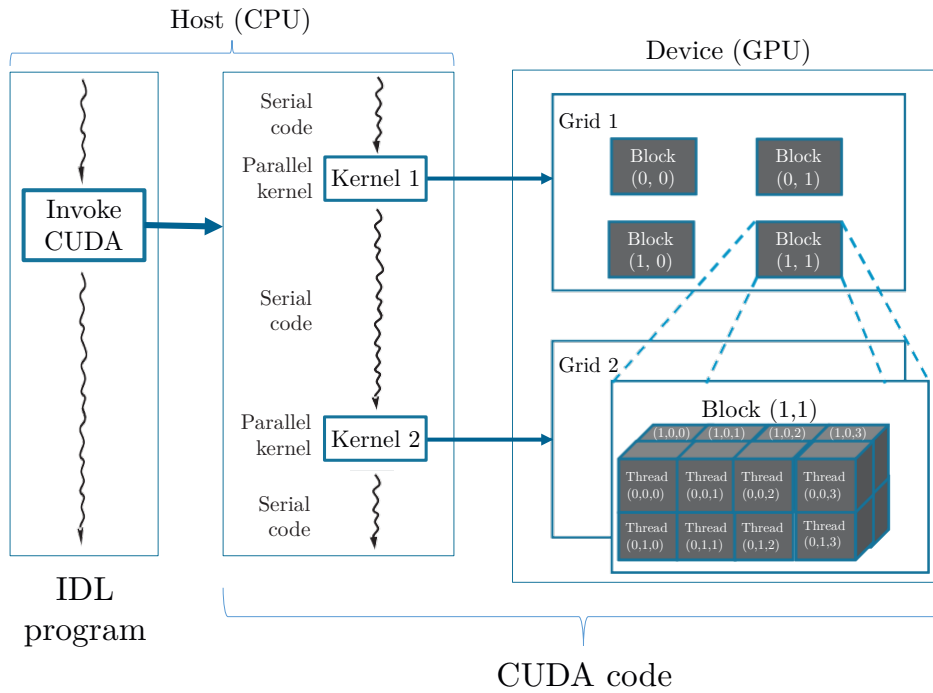


Figure 3.8: Execution of an IDL program invoking CUDA and using a multidimensional example of a CUDA grid organization.

3.4.1 Hardware & Software

The hardware used in this work is detailed in Tables 3.2 and 3.3. The devices of the two systems are both CUDA-enabled GPUs [150]. For more information concerning the devices' specifications refer to [151], [152].

Table 3.2: Systems used in this work.

	System 1	System 2
Operating System	CentOS 6.5	
CPU	Intel Xeon E5530 (4x 2.40 GHz)	Intel Xeon W3505 (2x 2.53 GHz)
Main Memory	11.6 GB	23.4 GB
GPU	Quadro FX 3800	Quadro 600
Device Drivers	331.62	

Table 3.3: Devices used in this work.

	Quadro FX 3800	Quadro 600
CUDA Cores	192	96
Memory Bandwidth (GB/s)	51.2	25.6
Memory (GB)	1	
Compute Capability	1.3	2.1
Microarchitecture	Tesla	Fermi
Maximum Power Consumption (W)	108	40

In this dissertation the 6.0 CUDA Toolkit version was used, in which the Thrust version 1.7.0 and a compiler for NVIDIA GPUs known as NVCC, which stands for NVIDIA's CUDA Compiler, are included. The IDL Version that was used was the 8.2.2 for System 1 and 8.0 for System 2. In order to visualize and study each reconstructed image, Quasi-Manager was used, an image visualization and data analysis software developed within the group.

In order to guarantee accurate execution time measurements, all the computer processes alien to the reconstruction process were kept to a minimum. Otherwise, the reconstruction times would increase.

3.4.2 CUDA Code

The implementation of the system matrix calculation needs to thoroughly address two major questions:

1. How to handle the huge amount of data regarding memory resources?
2. How to set its elements?

As stated in Sub-section 2.4.1, GPUs are specialized for intensive computation. However, their memory is quite limited. In fact, in this work the GPUs that were used only had 1 GB of memory each, leading to some memory constraints. In the medical imaging field 1 GB is far from enough to contain all the necessary data for image reconstruction due to the detectors' high resolution.

For the system matrix calculation of a single projection, it is necessary to have in memory a matrix that corresponds to the estimation of the attenuation coefficients of the object, sizing the dimension of the FOV ($Detector_{Dim_x} \times Detector_{Dim_y} \times NSlices$) times `sizeof(float)`, being `sizeof(float)` the dimension of a `float` variable which in memory corresponds to 4 bytes. Therefore, the size of this matrix can achieve approximately 2.26 GB ($2816 \times 3584 \times 60 \times 4 \text{ bytes} = 2,422,210,560 \text{ bytes}$). Note that this memory allocation is related to the system matrix for a single projection. For the simultaneous calculation of the 25 projections with this system matrix calculation approach it would be necessary to have more than 56 GB available, which is not viable as there are currently no commercially available GPUs close to this amount of memory. For the CUDA code, extra memory allocation is needed due to the use of the `sort_by_key` function of Thrust. However, even the 2.26 GB necessary memory allocation is more than twice the available 1 GB in each GPU, which is a matter that needs to be dealt with caution in order to successfully transfer data to the device.

The CUDA code implementation of the system matrix calculation copes with a single group of bins in order to address the memory limitations. This group possesses 88×56 bins, which means that it is necessary to compute multiple groups of bins in order to

achieve a complete system matrix for a single projection (2816×3584 bins). The maximum space that every step of the calculation could occupy was considered in the calculation of the GPU memory occupation. The general idea of the CUDA code is as follows: the determination of the intersections is parallelized by distributing the rays between available threads; for the calculation of the distances between intersections, each thread is responsible to calculate a distance between intersection n and $n + 1$.

The code was implemented in order to allow its incorporation within the pure-IDL program. Due to the length of the CUDA code, this sub-section does not intend to explain in depth how the program works. Instead, for a more detailed view of the program, a pseudocode (Listing 3.5) is presented which covers the key stages of the entire code.

Listing 3.5: Pseudocode of the CUDA code for the system matrix calculation.

```

1  // This code takes care of a single group of 88*56 detector bins (x*y)
2  foreach thread // Obtain the intersections, where each thread is associated
   ↪ with a single bin (x, y) = (thread_ID_x, thread_ID_y)
3  // Considering Slopevector = B-A in Equation 3.2
4  Slopevectorx = xfocus - (thread_ID_x+0.5)*xbinsize
5  Slopevectory = yfocus - (thread_ID_y+0.5)*ybinsize
6  Slopevectorz = zfocus - (distance_from_breast_support_table_to_detector)
7
8  // Determine ray xx intersections
9  foreach x_integer (thread_ID_x+1 -> detector_dim_x/2)
10   // Solve Equation 3.6
11   x = x_integer * xbinsize
12   alpha = (x - (thread_ID_x+0.5)*xbinsize) / Slopevectorx
13   // Determine corresponding y and z
14   y = (thread_ID_y+0.5)*ybinsize + alpha*Slopevectory
15   z = distance_from_breast_support_table_to_detector + alpha*Slopevectorz
16
17   Add the coordinates of the intersections (x, y and z) and the bin to an
   ↪ array (one array per coordinate)
18
19   Find ray yy intersections (analogous to xx intersections)
20   Find ray zz intersections (analogous to xx intersections)
21
22 // Deletes all intersections that occur outside the FOV
23 foreach intersection
24   if(x_intersection < 0 OR x_intersection > detector_dim_x/2 OR
   ↪ y_intersection < 0 OR y_intersection > detector_dim_y OR
   ↪ z_intersection < 0 OR z_intersection > NSlices) // NSlices
   ↪ corresponds to the number of slices, in this work 60
25   Exclude intersection
26
27 Group the intersections obtained with different axis - put all the x
   ↪ coordinates into one array and do the same for y and z coordinates
28 Sort intersections using the x-coordinate and then by bin
29

```

```

30  foreach thread // Calculate the distances, where each thread is responsible
    ↪ to calculate a distance between intersection n and n+1
31  // Calculate the Euclidean distance between each intersection:
32  distance = ||next_intersection - intersection||
33
34  // From the intersections' coordinates find the corresponding bin:
35  if (x_intersection == integer(x_intersection))
36      coordinate_voxel_x = x_intersection - 1
37  else
38      coordinate_voxel_x = floor(x_intersection)
39  Find coordinate_voxel_z (analogous to coordinate_voxel_x)
40  if (y_intersection < yfocus)
41      if (y_intersection == integer(y_intersection))
42          coordinate_voxel_y = y_intersection-1
43      else
44          coordinate_voxel_y = floor(y_intersection)
45  else
46      coordinate_voxel_y = floor(y_intersection)
47
48  // After these steps the system matrix is obtained, comprising 3 arrays:
    ↪ bins, voxels and distances

```

3.4.3 Invoke CUDA from IDL

IDL offers a few ways of interfacing with external programs. For this dissertation the options `CALL_EXTERNAL` [153] and `LINKIMAGE` [154] were studied in order to invoke CUDA from IDL. Despite `LINKIMAGE` being a more challenging and less simpler technique compared with `CALL_EXTERNAL` (in which developers do not need a broad understanding of IDL internals), it revealed to be a more resourceful method to link a CUDA program with IDL. It has the advantage of routines written in this format behave like built in IDL routines, requiring a separate `LINKIMAGE` call that must be done before any of the routines. Thus, `LINKIMAGE` demonstrated to require a steep learning curve.

For the first contact with `LINKIMAGE` and how to integrate an external program with IDL, the IDL-C interface [155] was explored by identifying the possibility to access and create IDL variables within the C module. In order to accomplish this, several tests with the C programming language started with simple programs while increasing their complexity — from a simple variable passed from IDL to the C module (to be changed and returned back to IDL), to complex matrices' operations to be completed in the C module and the respective results returned to the IDL program. From this learning experience, the key-aspects are:

- `LINKIMAGE` C modules must be compiled and linked before the IDL execution — the C module has to include the location of the "export.h" header file in the compilation step;
- every variable in IDL is in reality a C structure called `IDL_VARIABLE`;

- the access of IDL variables must be done through a C pointer called `IDL_VPTR`, in which the extracted values depend upon how it was created in IDL;
- in the data transfer between C and IDL one must consider that the representation of a data type may be different (e.g. integers in IDL are short C integers).

The CUDA invocation from IDL is considerably similar to what is done between IDL and the C programming language. The major difference is the compilation step, in which with CUDA a different compiler, NVCC, is used. Before adapting the CUDA code for the system matrix calculation to this scenario, several CUDA-IDL tests were performed. One of the most relevant tests, was when a CUDA code received three vectors, where the first two are filled with matrices' elements, and then computes on the last one a pair-wise addition and returns it back to the IDL program.

The CUDA code responsible for the system matrix calculation was afterwards incorporated into the already existent pure-IDL implementation with several modifications in both codes. In the IDL an external CUDA function call was included that is responsible for transferring to the CUDA code the necessary information to compute the system matrix calculation for a particular group of 88×56 detector bins. In turn, the CUDA code suffered modifications in order to accommodate the IDL-variables and to transfer back to IDL the computed part of the system matrix — voxels' coordinates and respective distances. Then, IDL processes that data and, when finished, proceeds to the next group of detector bins, repeating the same procedure several times until every bin in the left-half of the x -dimension of the detector is addressed.

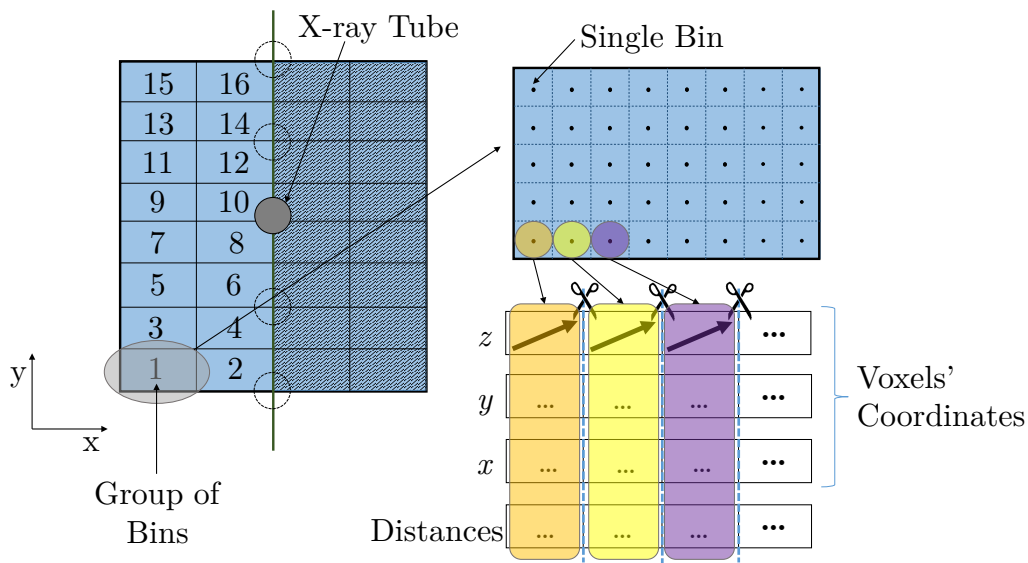


Figure 3.9: Illustration of how the data received in IDL from the CUDA code (the voxels' coordinates and distances) are processed to organize it per bin.

In order to process a group of 88×56 detector bins, the IDL receives the voxels' coordinates and respective distances from the CUDA code and applies the following instructions (Figure 3.9): check the z -coordinate of each two voxels and verify if they change from 60 (the maximum z) to 0 (the minimum z); if so, the algorithm cuts between those elements, grouping the voxels that correspond to the same bin. Finally, every voxel that corresponds to a single-bin is processed with an iterative reconstruction algorithm, in the same way as it would be done in the pure-IDL version. Note that this CUDA-IDL implementation uses a scale factor in order to have faster reconstructions, being considered that each bin possesses a dimension of 4, 8 or 16 times the original 0.085 mm. This scaling results in a reduction of the computational resources involved.

3.4.4 Evaluation of the CUDA-IDL Implementation

In order to check the efficiency of the new CUDA-IDL implementation it is necessary to assess the quality of the results. To compare the overall results arising from the CUDA-IDL implementation and the pure-IDL implementation, two different metrics were considered: the Normalized Mean Error (NME) [156] and the speedup.

NME measures the difference between two images and is defined as the sum, for all the voxels in the image, of the absolute difference between the image voxel j obtained with the CUDA-IDL implementation (P_j) and the corresponding voxel in the image obtained with the pure-IDL implementation (O_j), normalized to the sum of all O_j :

$$NME = \frac{\sum_j |P_j - O_j|}{\sum_j O_j} \quad (3.7)$$

In this work NME is calculated with MATLAB^{®11} R2013b (version 8.2).

The speedup is calculated through the ratio of the computation time needed to complete reconstruction in both implementations:

$$speedup = \frac{time_{pure-IDL}}{time_{CUDA-IDL}} \quad (3.8)$$

¹¹MATLAB is a registered trademark of The MathWorks, Inc.

4

Results and Discussion

This chapter is intended to present the relevant results of this work, which aims to accelerate the system matrix calculation process. First, in Section 4.1, a motivation for the calculation of the system matrix using CUDA is given through the verification of how computationally intensive is this calculation in the whole reconstruction process. Then, Section 4.2 shows preliminary results of the CUDA integration in IDL using a single-thread per kernel strategy. The chapter ends in Section 4.3 with the test of the full integration, which uses a multi-threaded scheme per kernel.

4.1 Expended Time in the System Matrix Calculation

The calculation of the system matrix is the most computationally intensive part of the whole reconstruction process. In this section, the pure-IDL implementation was executed to show how computationally intensive is the system matrix calculation. This test was performed in two different systems (whose characteristics are presented in Table 3.2) for SART, ML-EM and OS-EM iterative algorithms. Particularly, SART was performed with a relaxation parameter, λ , of 0.2, and OS-EM with four different subsets (with image updates at the 6th, 12th, 18th and 25th projections). Three different bins' scale factors were also considered so that each bin corresponds to a dimension of 4, 8 or 16 times the original 0.085 mm. Scales of 2 and 1 (original size of the bin) were not tested because the reconstruction would be very time-consuming. For each combination of system, algorithm and scale, the procedure was repeated five times. The execution times of the full reconstructions were recorded, as well as the time spent in the system matrix calculation in each reconstruction. The average percentages of the entire computation of a single iteration spent in the calculation of the system matrix are provided in Table 4.1.

Table 4.1: Percentages of the entire computation of a single iteration spent in the calculation of the system matrix. These values are obtained from the pure-IDL implementation performed in two different systems for SART, ML-EM and OS-EM iterative algorithms, with three different bins' scale factors. Each presented value is an average of five different tests. The standard deviation of each value is not higher than 0.2%.

	Scale	SART ($\lambda = 0.2$)	ML-EM	OS-EM (4 subsets)
System 1	4	88.5%	89.1%	89.1%
	8	82.9%	83.4%	83.5%
	16	78.6%	79.5%	79.3%
System 2	4	84.6%	84.9%	84.8%
	8	80.5%	81.0%	80.9%
	16	76.5%	76.9%	76.9%

The results show that the lower the scale factor (i.e. the higher the amount of data to process), the higher the time spent in the system matrix calculation. For instance, with a scale factor of 4 with System 1, the time spent in the system matrix calculation is approximately 89% of the total, with a lower scale factor this percentage would be even higher. Comparing both systems, it is possible to verify that System 1 spends a higher percentage of the total time in this task. In fact, during these tests System 1 turned out to be slower than System 2 in the overall process of the reconstruction, which is a result of the different CPUs and available amount of memory.

Another test conducted in this work evaluated whether the multi-threading capability of IDL is being used in the pure-IDL implementation in both multi-core systems. For this purpose, the multi-threading functionality was disabled by setting the number of threads to use to one. This test revealed equal execution times compared to the ones previously determined, which indicates that the pure-IDL implementation does not make use of the multi-threading capability of IDL.

To sum up, the results of this section demonstrate that a promising way of significantly accelerate the image reconstruction process can be accomplished by focusing on reducing the system matrix calculation time.

4.2 Single-Thread Execution per Kernel Strategy

A first test of the CUDA integration in IDL was performed using a single-thread execution per kernel method. The determination of the system matrix is then completed a bin at a time (similarly to the pure-IDL implementation), in which a single ray created from a bin is processed by a single thread. This test can be seen as a sequential implementation in GPU.

The test performed in this section focused on verifying the performance of the CUDA-IDL integration by calculating the NME between the pure-IDL implementation and the

CUDA-IDL implementation. As in the previous section, this test was performed in the two available systems for SART, ML-EM and OS-EM iterative algorithms. For this test, a single bins' scale factor of 16 was used since it is extremely time-consuming to test an implementation that uses the GPU with only one thread per kernel — the host is always transferring information in and out from the device to process each ray. The NME values are shown in Table 4.2.

Table 4.2: NME percentages of a single iteration obtained through the comparison of the pure-IDL implementation with the CUDA-IDL implementation using a single thread CUDA kernel. This test was performed in two different systems for SART, ML-EM and OS-EM iterative algorithms and with a bins' scale factor of 16.

	SART ($\lambda = 0.2$)	ML-EM	OS-EM (4 subsets)
System 1			
System 2	0.05%	0.06%	0.18%

The results in the table show that both systems performed equally during the program execution. This means that the execution is independent from the system, being expected that the results are reproducible between systems with the minimum necessary requirements. Moreover, from the NME results it is possible to conclude that the reconstructed images obtained from SART and ML-EM are almost identical when comparing CUDA-IDL implementation against pure-IDL implementation. The NME values for the OS-EM algorithm with 4 subsets the NME is of one order of magnitude higher.

Following NME calculations, the differences between the images obtained with the pure-IDL implementation and the CUDA-IDL implementation are analysed. In Figure 4.1 the image results for the 27th z -plane are provided for the three different algorithms for System 1 (equal results are obtained with System 2). Comparing the images obtained from both implementations it is not possible to identify any significant visible difference. However, the data range of the difference images is 11 orders of magnitude lower than the reconstructed images for SART and ML-EM, and 10 for OS-EM. The presence of small disparities in every reconstructed image was already expected since the implementation of the CUDA code is not a direct translation of the pure-IDL code. All the other z -plane images show similar results concerning the order of magnitude and the location of the small artefacts. Note that these images do not represent their true value as a medical diagnostic tool, having a scale factor that considerably blurs the images.

The small artefacts seen in the difference images are more relevant to analyse in the OS-EM algorithm due to its higher NME value. Based on these results, a study of the influence of the number of subsets in the performance of the CUDA-IDL implementation was conducted for this algorithm. The NME percentages were calculated for different number of subsets (particularly, 1, 2, 4, 8 and 25) for a single iteration for System 1 with a scale factor of 16. Note that the case of OS-EM with a single subset corresponds to ML-EM. From Figure 4.2 it is possible to conclude that the higher the number of subsets, the

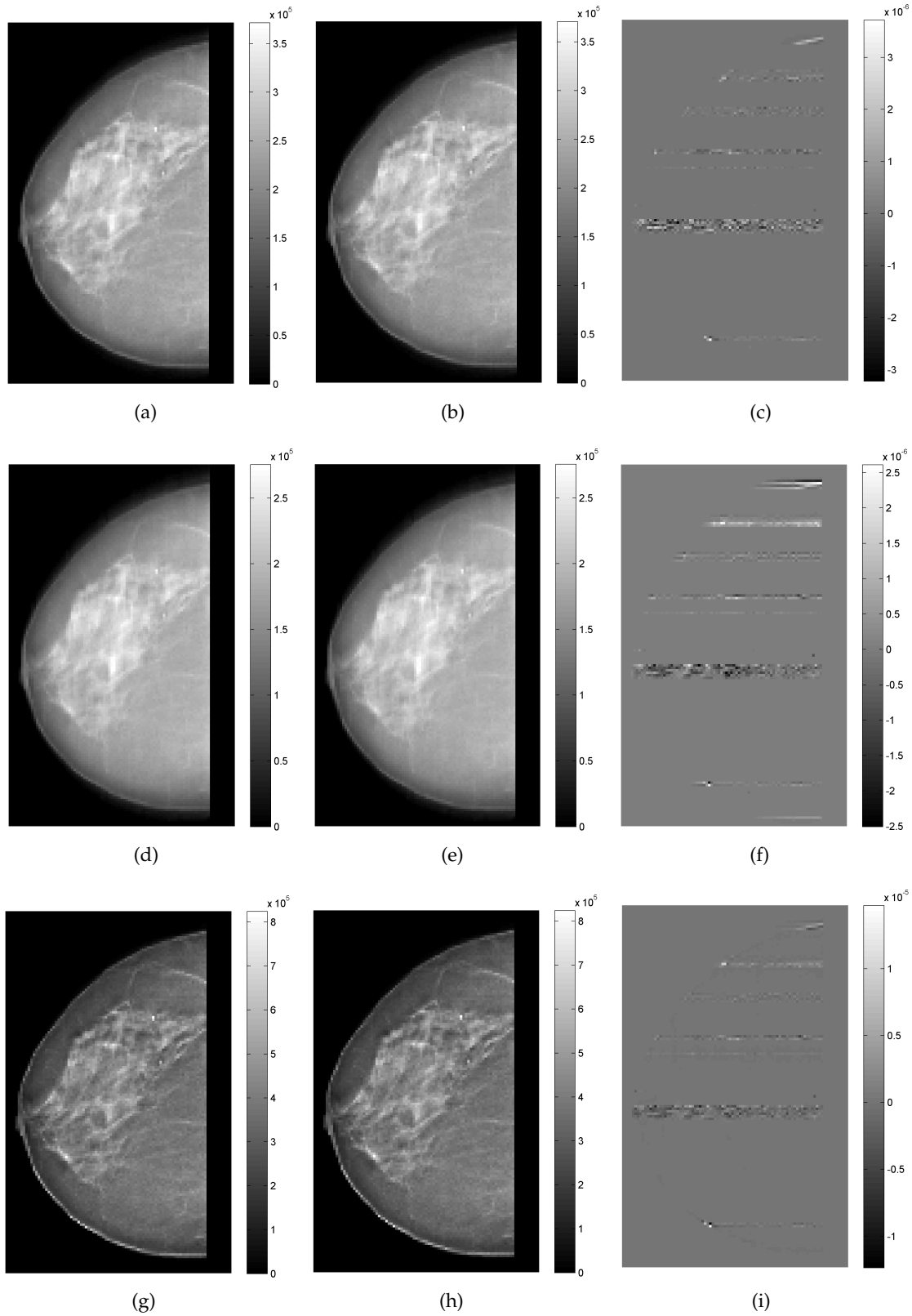


Figure 4.1: Reconstructed images of the 27th z -plane for a single iteration for System 1 with a bins' scale factor of 16 using (a)–(c) SART, (d)–(f) ML-EM and (g)–(i) OS-EM. Three different images per algorithm are shown, from left to right: CUDA-IDL implementation using a single-thread execution per kernel, pure-IDL implementation and difference between both.

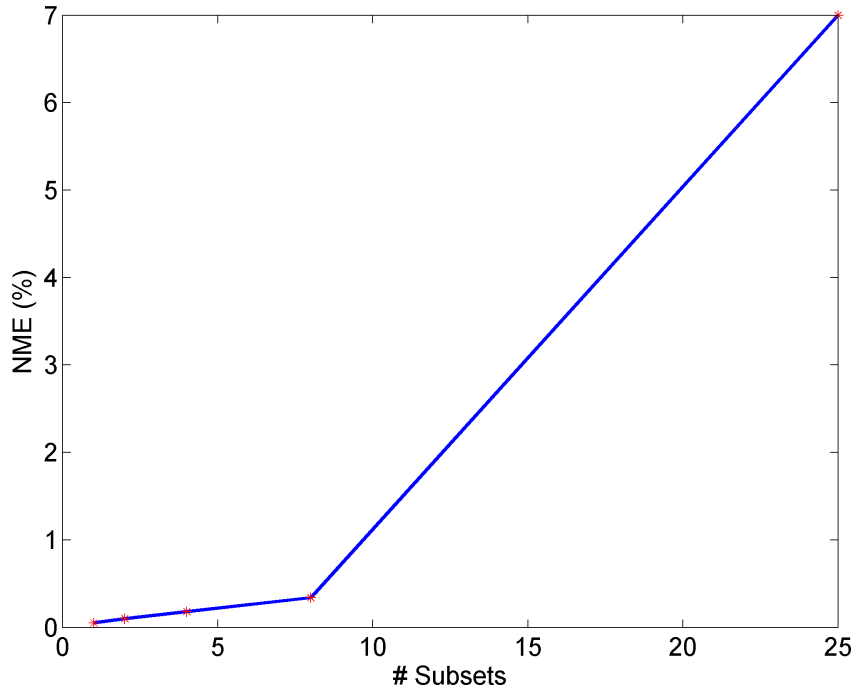


Figure 4.2: The NME percentages of the 3D DBT reconstructed image with OS-EM as a function of the number of subsets for a single iteration for System 1 with a bins' scale factor of 16.

higher the NME value. This is probably due to the fact that the image is being updated n times, being n the number of subsets. As there are already some small artefacts in the first update, these artefacts are being continuously accumulated.

The artefacts presented in the images seem to have a pattern and they appear in similar locations between images. It was further concluded through debugging that wrong results appeared after the invocation of some specific Thrust functions. However, it was not feasible to implement them from scratch in due time.

The following tests did not include OS-EM, as it was seen in this section that the artefacts with this algorithm are more significant. It was therefore decided to focus on the complete integration of SART and ML-EM.

4.3 Full Integration of CUDA in IDL

This final section deals with the complete CUDA-IDL integration, in which a standard multi-threaded kernel was used. In order to accomplish this, IDL was adapted to make use of the algorithm explained in Figure 3.9, in which IDL receives a large block of data from CUDA and organizes the information for further processing.

The goal of this work is to accelerate the reconstruction process through the use of

a GPU. Therefore, speedup is the most relevant metric to compare the pure-IDL implementation with the complete integration. In order to obtain this metric, the execution times for both implementations were measured. An average of computation times is presented in Table 4.3 for a single iteration for the two available systems with a bin's scale factor of 16 and with SART and ML-EM. These tests were performed five times for each combination of system, scale and algorithm, in order to have more statistical relevant results.

Table 4.3: Average of five execution times of the pure-IDL and the CUDA-IDL implementations and resulting speedup. These tests were completed for a single iteration of SART and ML-EM for the two available systems and for a bins' scale factor of 16. The standard deviation of both implementations do not exceed 4 s and 0.04 regarding the execution times and the speedups, respectively.

	Algorithm	Pure-IDL Time (s)	CUDA-IDL Time (s)	speedup
System 1	SART	165	110	1.50
	ML-EM	169	107	1.58
System 2	SART	150	111	1.35
	ML-EM	154	103	1.50

As seen in the table, the pure-IDL reconstruction times for System 2 are slightly faster than for System 1. As for the CUDA-IDL implementation, the execution times are similar. The speedup obtained for ML-EM is slightly higher in both systems when comparing with the performance of SART, achieving 1.6 for System 1. This speedup can be largely amplified with the use of a smaller bin's scale factor since, as seen in Section 4.1, the lower the scale the greater the time of the entire computation spent in the system matrix calculation is, with a corresponding higher use of the GPU.

The GPU used to run the CUDA part has a significant influence on the speedup that can be obtained. The selection of a new GPU was also pursued during this dissertation aiming to further accelerate the reconstruction process. The selected GPU card is similar to NVIDIA GeForce GTX 780 [157], which has 3 times more memory than the GPUs used in this dissertation, and will be used in future work. This will enable to reduce the sequential portion of the IDL code through the increase of the number of bins to be processed per CUDA invocation (reducing the amount of CUDA invocations). This will result in a sharper increase of the speedup because even a small reduction of the sequential part of the code has a large impact in the speedup according to the Amdahl's law (Equation 2.10). The speedup achieved with this methodology (CUDA integration in an IDL reconstruction process for DBT) cannot be easily compared with the literature since this methodology was applied for the first time.

The NME was calculated and the results are presented in Table 4.4. These results are three orders of magnitude higher than those of Table 4.2, revealing that there is room for improvements, mainly in the CUDA code (CUDA does not provide the needed IDL

Table 4.4: NME percentages of a single iteration obtained through the comparison of the pure-IDL implementation with the CUDA-IDL implementation using the standard multiple thread kernel. This test was performed in two different systems for SART and ML-EM iterative algorithms and with a bins' scale factor of 16.

	SART ($\lambda = 0.2$)	ML-EM
System 1	24.70%	22.02%
System 2	24.75%	22.00%

optimized functions and they must be programmed from scratch). These NME values are in agreement with the resulting images (Figure 4.3) and the cause for these errors was investigated through several tools, including the CUDA debugger. It was identified that several times the IDL component was receiving an amount of data different from the expected revealing amplified artefacts when compared with the ones seen in the previous section. In fact, IDL was supposed to receive data concerning 88×56 bins, however it was receiving data that corresponded most of the times to a different number of bins. The above table also reveals that there are small differences concerning the NME values for the two systems, which may be due to the different IDL versions installed in each system.

The goal of this dissertation was mainly to prove the concept that the acceleration of this process is possible through the use of GPUs. In order to maintain the image quality achieved with the pure-IDL implementation, the removal of the previously mentioned artefacts that are originated from the CUDA code is mandatory.

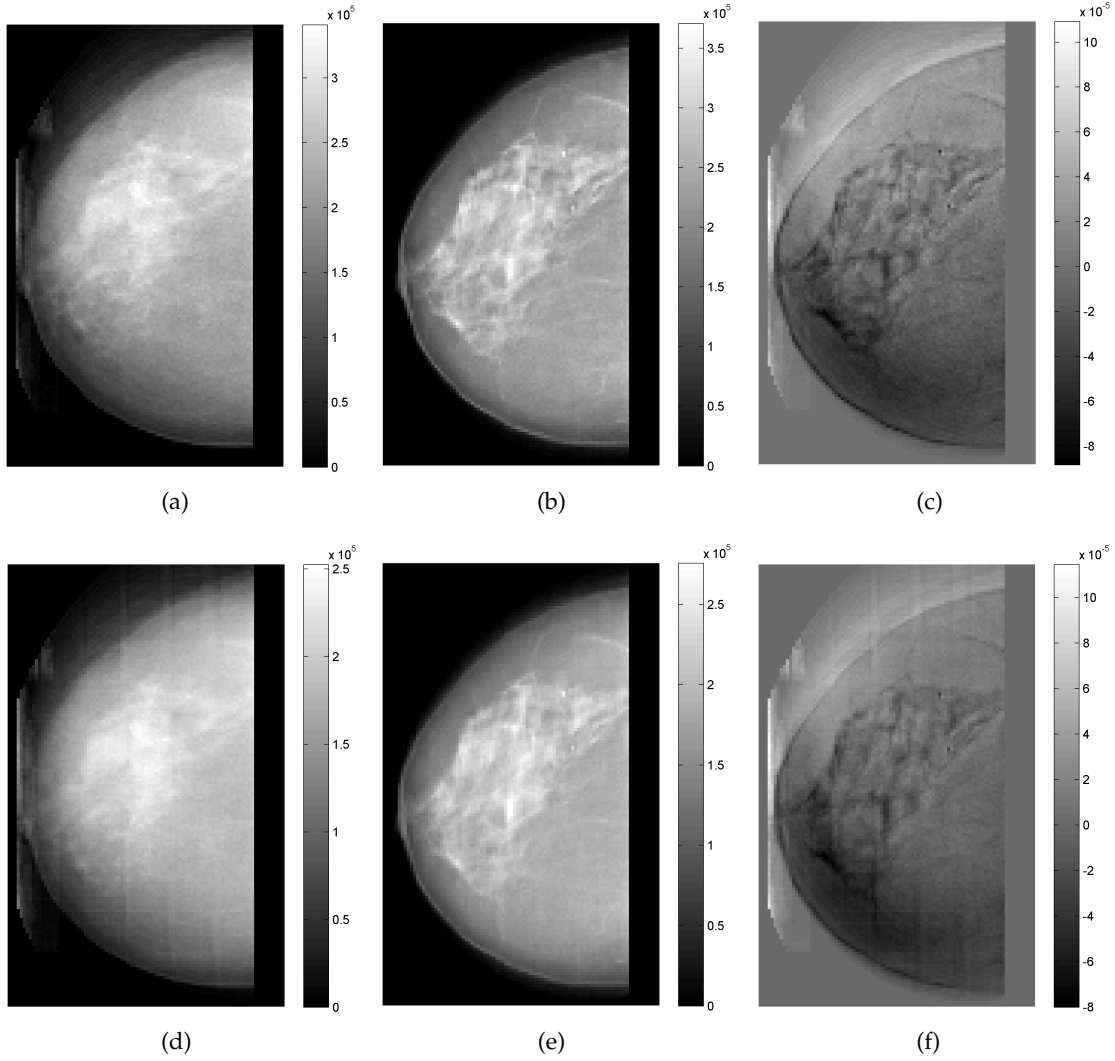


Figure 4.3: Reconstructed images of the 27th z -plane for a single iteration for System 1 with a bins' scale factor of 16 using (a)–(c) SART and (d)–(f) ML-EM. Three different images per algorithm are shown, from left to right: CUDA-IDL implementation using a multi-thread scheme per kernel call, pure-IDL implementation and difference between both.

Conclusions and Future Work

Iterative algorithms have been shown to produce the highest quality DBT images but are computationally intensive which leads to the rejection of their clinical use. This work aimed to accelerate the DBT image reconstruction process for three of these algorithms (SART, ML-EM and OS-EM) on GPUs using CUDA. The use of GPUs for reconstruction represents a technical breakthrough and for this work the followed methodology consisted in integrating CUDA with an already existent implementation in IDL within the research group, aiming to obtain a scalable application. To the best of the authors' knowledge this method has never been attempted before for DBT.

The developed CUDA-IDL implementation makes use of a CUDA program that provides the calculation of the system matrix, which was shown in this work to be the most computationally expensive part of the reconstruction, being a key point to be parallelized. The integration between IDL and CUDA allows an incremental development of the image reconstruction program by replacing sequential modules in IDL by parallelized ones as soon as they are available. The parallelization of other parts is future work, mainly the forward projection and backprojection steps (Figure 3.3).

The integration of CUDA in IDL was successfully accomplished with promising speedups. For SART and MLEM speedups up to 1.6 were obtained which prompt to use GPUs in a near future in a real clinical setting. The developed implementation proved to be computationally efficient, allowing to reduce computation times when compared with the pure-IDL implementation.

This work is part of a large-scale two-year research project, developed between multiple institutions (Figure 5.1), which is concluding its first year. This project aims to improve the quality of image and reduce the dose in DBT through the use of statistical algorithms. The developed CUDA-IDL implementation is a key component for this project

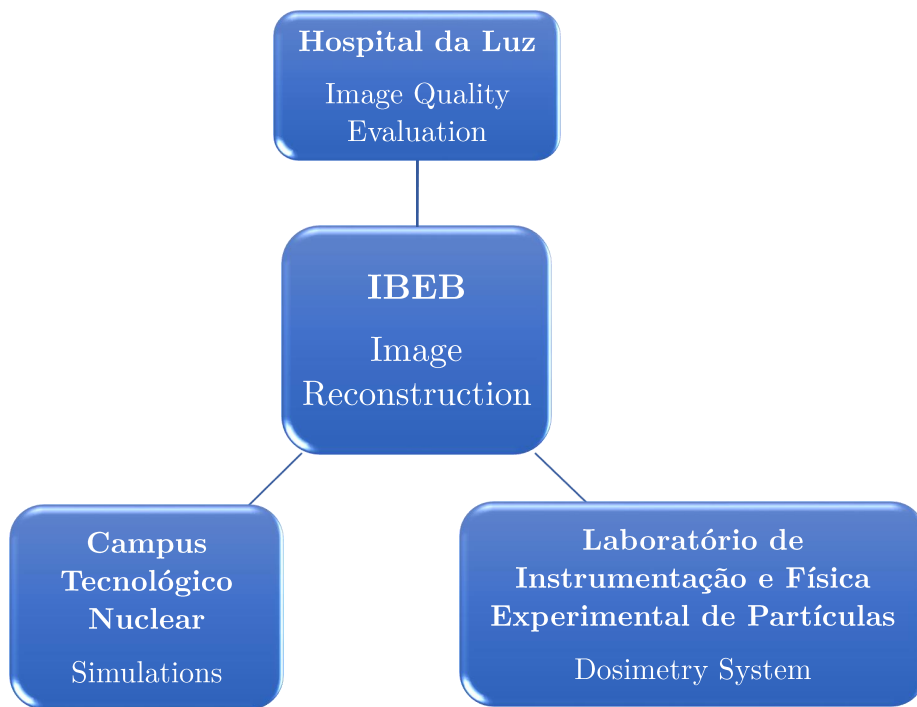


Figure 5.1: Larger project entitled "Improvements of image quality and dose reduction in digital breast tomosynthesis using statistical image reconstruction algorithms" in which this dissertation is integrated.

in order to achieve a fast process with turnaround times compatible with its use in a real clinical setting. In this dissertation, a new GPU was selected to be used in the continuation of this larger project, in order to further accelerate the image reconstruction process.

At the present state of this work, the system matrix calculation is performed on the fly due to memory constraints. With the new GPU which has more memory, it may be possible to calculate entirely the system matrix before the image reconstruction process; comparisons between this new strategy and the current one should be made. The pre-calculation provides faster reconstructions as the most time-consuming process is done before experiments. On the other hand, the calculation on the fly provides more flexibility.

In order to use the methodology of this work in a real clinical setting, improvements in the CUDA program have to be done in order to obtain artefact-free images. Furthermore, the speedups that were obtained must be increased through several possibilities:

- **More GPU memory** — the GPUs used in this work with 1 GB are not enough to parallelize all the data that is needed to calculate the system matrix; besides, Amdahl's law (Equation 2.10) shows that the smallest reduction in the amount of sequential code has extreme importance in enhancing speedup; moreover, the current state of the application executes several transfers between CPU and GPU, which must be minimized in order to further reduce execution times.

- **Smaller bin's scale factor** — this work showed that the use of a smaller bin's scale factor contributes to a more efficient use of the GPU capabilities, which also results in an improvement of the resolution of the 3D reconstructed image.
- **Profile the CUDA code and monitoring GPU occupancy** — both aspects can help to spot the code segments where modifications will bring significant performance enhancements.
- **Unified Memory in CUDA 6** — CUDA programming is a field that is changing day by day and during this dissertation the Unified Memory concept emerged [158], which will largely simplify the complexity of CUDA programs, namely the system matrix calculation.
- **Multiple GPUs** — the use of multiple GPUs creates further opportunities for making in parallel different parts of the global calculation.

CUDA programming is very hard, even with the several existent libraries. In this work the Thrust library was used, which definitely enhanced the productivity by providing automatic mappings of computational tasks onto the GPU. A very significant part of the work conducted throughout this dissertation was focused on learning a new programming paradigm (parallel programming), which is less intuitive than single-threaded programming, and on developing an appropriate methodology. The most significant disadvantage of GPU programming when compared with sequential programming is the longer development cycle, either by the difficulty in obtaining correct code either by the time it takes to enhance its performance. The available debuggers are, most of the times, non user-friendly which makes the task even harder.

The work targeted at developing parallel versions of DBT-based image reconstruction can evolve in several directions. One possibility is the use of different types of accelerators like AMD^{®12} GPUs [159] or Intel[®] Xeon Phi^{™13} [160]: this dimension will demand an alternative to CUDA that only targets NVIDIA GPUs, namely OpenCL [122]. Another direction that can be followed is the use of parallel programming frameworks like OpenMP^{®14} [161] and OpenACC [162].

The developed program makes use of CPUs and GPUs in a parallel manner to maximize the utilization of computer resources. The results shown in this dissertation are very promising and may lead to the reduction of radiation dose received by patients during a DBT scan and also may lead to guided biopsies. The methodology of this work may be adapted to other medical imaging modalities such as CT and MRI, so that GPUs become a gold standard in the field of medical image reconstruction, where large amounts of data need to be processed.

¹²AMD is a registered trademark of Advanced Micro Devices, Inc.

¹³Intel and Xeon Phi are trademarks of Intel Corporation.

¹⁴OpenMP is a registered trademark of the OpenMP Architecture Review Board in the United States and other countries.

5.1 Contributions

The contributions of this dissertation are as follows:

- Poster presentation in the 6th Workshop on Biomedical Engineering (WBME), held in Lisbon, Portugal, on the 5th April 2014 — Appendix [A](#);
- Poster presentation in the Workshop MIEI 2014, held in Caparica, Portugal, on the 27th June 2014;
- Award for the best oral presentation in the Workshop MIEI 2014, held in Caparica, Portugal, on the 27th June 2014;
- Participation and poster presentation in the 5th edition of the Programming and Tuning Massively Parallel Systems summer school (PUMPS), held in Barcelona, Spain, from the 7th to 11th July 2014 — Appendix [B](#);
- Participation in the 5th edition of the Fraunhofer Portugal Challenge 2014 (approved for 2nd phase);
- Participation in the Concurso Nacional de Inovação BES 2014 (results pending).
- Development of a quick guide entitled "CUDA-IDL Implementation of Digital Breast Tomosynthesis Image Reconstruction" for Users and Future Developers — Appendix [C](#);

Further work is being prepared for submission in conferences and possibly a journal paper.

Bibliography

- [1] R. Siegel, J. Ma, Z. Zou, and A. Jemal, "Cancer statistics, 2014", *CA: A Cancer Journal for Clinicians*, vol. 64, no. 1, pp. 9–29, 2014.
- [2] J. Ferlay, E. Steliarova-Foucher, J. Lortet-Tieulent, S. Rosso, J. W. W. Coebergh, H. Comber, D. Forman, and F. Bray, "Cancer incidence and mortality patterns in europe: estimates for 40 countries in 2012", *European Journal of Cancer*, vol. 49, no. 6, pp. 1374–1403, 2013.
- [3] F. P. Turkoz, M. Solak, I. Petekkaya, O. Keskin, N. Kertmen, F. Sarici, Z. Arik, T. Babacan, Y. Ozisik, and K. Altundag, "Association between common risk factors and molecular subtypes in breast cancer patients", *The Breast*, vol. 22, no. 3, pp. 344–350, 2013.
- [4] B. S. Hulka and P. G. Moorman, "Breast cancer: hormones and other risk factors", *Maturitas*, vol. 38, no. 1, pp. 103–113, 2001.
- [5] D. W. Thompson, "Genetic epidemiology of breast cancer", *Cancer*, vol. 74, no. S1, pp. 279–287, 1994.
- [6] Collaborative Group on Hormonal Factors in Breast Cancer, "Familial breast cancer: collaborative reanalysis of individual data from 52 epidemiological studies including 58 209 women with breast cancer and 101 986 women without the disease", *The Lancet*, vol. 358, no. 9291, pp. 1389–1399, 2001.
- [7] C. S. Berkey, A. L. Frazier, J. D. Gardner, and G. A. Colditz, "Adolescence and breast carcinoma risk", *Cancer*, vol. 85, no. 11, pp. 2400–2409, 1999.
- [8] J. L. Kelsey, M. D. Gammon, and E. M. John, "Reproductive factors and breast cancer", *Epidemiologic reviews*, vol. 15, no. 1, pp. 36–47, 1993.
- [9] L. Titus-Ernstoff, M. P. Longnecker, P. A. Newcomb, B. Dain, E. R. Greenberg, R. Mittendorf, M. Stampfer, and W. Willett, "Menstrual factors in relation to breast cancer risk", *Cancer Epidemiology Biomarkers & Prevention*, vol. 7, no. 9, pp. 783–789, 1998.

- [10] Collaborative Group on Hormonal Factors in Breast Cancer, "Breast cancer and hormonal contraceptives: collaborative reanalysis of individual data on 53 297 women with breast cancer and 100 239 women without breast cancer from 54 epidemiological studies", *Lancet*, vol. 347, no. 9017, pp. 1713–1727, 1996.
- [11] N. Biglia, E. Defabiani, R. Ponzzone, L. Mariani, D. Marengo, and P. Sismondi, "Management of risk of breast carcinoma in postmenopausal women", *Endocrine-Related Cancer*, vol. 11, no. 1, pp. 69–83, 2004.
- [12] D. R. Pathak, J. R. Osuch, and J. He, "Breast carcinoma etiology", *Cancer*, vol. 88, no. S5, pp. 1230–1238, 2000.
- [13] Collaborative Group on Hormonal Factors in Breast Cancer, "Breast cancer and hormone replacement therapy: collaborative reanalysis of data from 51 epidemiological studies of 52 705 women with breast cancer and 108 411 women without breast cancer", *The Lancet*, vol. 350, no. 9084, pp. 1047–1059, 1997.
- [14] D. M. Parkin, "International variation", *Oncogene*, vol. 23, no. 38, pp. 6329–6340, 2004.
- [15] D. M. Parkin, P. Pisani, and J. Ferlay, "Global cancer statistics", *CA: A Cancer Journal for Clinicians*, vol. 49, no. 1, pp. 33–64, 1999.
- [16] R. G. Ziegler, R. N. Hoover, M. C. Pike, A. Hildesheim, A. M. Y. Nomura, D. W. West, A. H. Wu-Williams, L. N. Kolonel, P. L. Horn-Ross, J. F. Rosenthal, and M. B. Hyer, "Migration patterns and breast cancer risk in asian-american women", *Journal of the National Cancer Institute*, vol. 85, no. 22, pp. 1819–1827, 1993.
- [17] S. A. Robert, A. Trentham-Dietz, J. M. Hampton, J. A. McElroy, P. A. Newcomb, and P. L. Remington, "Socioeconomic risk factors for breast cancer: distinguishing individual- and community-level effects", *Epidemiology*, vol. 15, no. 4, pp. 442–450, 2004.
- [18] N. G. Hildreth, R. E. Shore, and P. M. Dvoretzky, "The risk of breast cancer after irradiation of the thymus in infancy", *New England Journal of Medicine*, vol. 321, no. 19, pp. 1281–1284, 1989.
- [19] A. B. Miller, G. R. Howe, G. J. Sherman, J. P. Lindsay, M. J. Yaffe, P. J. Dinner, H. A. Risch, and D. L. Preston, "Mortality from breast cancer after irradiation during fluoroscopic examinations in patients being treated for tuberculosis", *New England Journal of Medicine*, vol. 321, no. 19, pp. 1285–1289, 1989.
- [20] M. J. Yaffe and J. G. Mainprize, "Risk of radiation-induced breast cancer from mammographic screening", *Radiology*, vol. 258, no. 1, pp. 98–105, 2011.
- [21] V. Chajès and I. Romieu, "Nutrition and breast cancer", *Maturitas*, vol. 77, no. 1, pp. 7–11, 2014.

- [22] W. Zheng, D. R. Gustafson, D. Moore, C.-P. Hong, K. E. Anderson, L. H. Kushi, T. A. Sellers, A. R. Folsom, R. Sinha, and J. R. Cerhan, "Well-done meat intake and the risk of breast cancer", *Journal of the National Cancer Institute*, vol. 90, no. 22, pp. 1724–1729, 1998.
- [23] M. A. S. V. Duyn and E. Pivonka, "Overview of the health benefits of fruit and vegetable consumption for the dietetics professional: selected literature", *Journal of the American Dietetic Association*, vol. 100, no. 12, pp. 1511–1521, 2000.
- [24] I.-M. Lee, "Antioxidant vitamins in the prevention of cancer", *Proceedings of the Association of American Physicians*, vol. 111, no. 1, pp. 10–15, 1999.
- [25] W. Y. Chen, B. Rosner, S. E. Hankinson, G. A. Colditz, and W. C. Willett, "Moderate alcohol consumption during adult life, drinking patterns, and breast cancer risk", *JAMA*, vol. 306, no. 17, pp. 1884–1890, 2011.
- [26] K. W. Singletary and S. M. Gapstur, "Alcohol and breast cancer: review of epidemiologic and experimental evidence and potential mechanisms", *JAMA*, vol. 286, no. 17, pp. 2143–2151, 2001.
- [27] S. A. Smith-Warner, D. Spiegelman, S.-S. Yaun, P. A. van den Brandt, A. R. Folsom, R. A. Goldbohm, S. Graham, L. Holmberg, G. R. Howe, J. R. Marshall, *et al.*, "Alcohol and breast cancer in women: a pooled analysis of cohort studies", *JAMA*, vol. 279, no. 7, pp. 535–540, 1998.
- [28] Z. Huang, S. E. Hankinson, G. A. Colditz, M. J. Stampfer, D. J. Hunter, J. E. Manson, C. H. Hennekens, B. Rosner, F. E. Speizer, and W. C. Willett, "Dual effects of weight and weight gain on breast cancer risk", *JAMA*, vol. 278, no. 17, pp. 1407–1411, 1997.
- [29] R. E. Harris, K. K. Namboodiri, and E. L. Wynder, "Breast cancer risk: effects of estrogen replacement therapy and body mass", *Journal of the National Cancer Institute*, vol. 84, no. 20, pp. 1575–1582, 1992.
- [30] A. Trentham-Dietz, P. A. Newcomb, K. M. Egan, L. Titus-Ernstoff, J. A. Baron, B. E. Storer, M. Stampfer, and W. C. Willett, "Weight change and risk of postmenopausal breast cancer (United States)", *Cancer Causes & Control*, vol. 11, no. 6, pp. 533–542, 2000.
- [31] P. H. Lahmann, K. Hoffmann, N. Allen, *et al.*, "Body size and breast cancer risk: findings from the european prospective investigation into cancer and nutrition (EPIC)", *International Journal of Cancer*, vol. 111, no. 5, pp. 762–771, 2004.
- [32] C. Héry, J. Ferlay, M. Boniol, and P. Autier, "Quantification of changes in breast cancer incidence and mortality since 1990 in 35 countries with caucasian-majority populations", *Annals of Oncology*, vol. 19, no. 6, pp. 1187–1194, 2008.

- [33] H. E. Karim-Kos, E. de Vries, I. Soerjomataram, V. Lemmens, S. Siesling, and J. W. W. Coebergh, "Recent trends of cancer in europe: a combined approach of incidence, survival and mortality for 17 cancer sites since the 1990s", *European Journal of Cancer*, vol. 44, no. 10, pp. 1345–1389, 2008, Cancer Control in Europe: State of the Art in 2008.
- [34] C. Hermon and V. Beral, "Breast cancer mortality rates are levelling off or beginning to decline in many western countries: analysis of time trends, age-cohort and age-period models of breast cancer mortality in 20 countries", *British journal of cancer*, vol. 73, no. 7, pp. 955–960, 1996.
- [35] A. Jemal, M. M. Center, C. DeSantis, and E. M. Ward, "Global patterns of cancer incidence and mortality rates and trends", *Cancer Epidemiology Biomarkers & Prevention*, vol. 19, no. 8, pp. 1893–1907, 2010.
- [36] M. L. Brown, F. Houn, E. A. Sickles, and L. G. Kessler, "Screening mammography in community practice: positive predictive value of abnormal findings and yield of follow-up diagnostic procedures", *AJR. American Journal of Roentgenology*, vol. 165, no. 6, pp. 1373–1377, 1995.
- [37] M. G. Wallis, M. T. Walsh, and J. R. Lee, "A review of false negative mammography in a symptomatic population", *Clinical Radiology*, vol. 44, no. 1, pp. 13–15, 1991.
- [38] H. Joensuu, R. Asola, K. Holli, E. Kumpulainen, V. Nikkanen, and L.-M. Parvinen, "Delayed diagnosis and large size of breast cancer after a false negative mammogram", *European Journal of Cancer*, vol. 30, no. 9, pp. 1299–1302, 1994.
- [39] C. Waldherr, P. Cerny, H. J. Altermatt, G. Berclaz, M. Ciriolo, K. Buser, and M. J. Sonnenschein, "Value of one-view breast tomosynthesis versus two-view mammography in diagnostic workup of women with clinical signs and symptoms and in women recalled from screening", *AJR. American Journal of Roentgenology*, vol. 200, no. 1, pp. 226–231, 2013.
- [40] J. G. Elmore, M. B. Barton, V. M. Moceris, S. Polk, P. J. Arena, and S. W. Fletcher, "Ten-year risk of false positive screening mammograms and clinical breast examinations", *New England Journal of Medicine*, vol. 338, no. 16, pp. 1089–1096, 1998.
- [41] R. A. Hubbard, K. Kerlikowske, C. I. Flowers, B. C. Yankaskas, W. Zhu, and D. L. Miglioretti, "Cumulative probability of false-positive recall or biopsy recommendation after 10 years of screening mammography: a cohort study", *Annals of Internal Medicine*, vol. 155, no. 8, pp. 481–492, 2011.
- [42] J. M. Park, E. A. Franken, M. Garg, L. L. Fajardo, and L. T. Niklason, "Breast tomosynthesis: present considerations and future applications", *Radiographics*, vol. 27, no. Suppl 1, S231–S240, 2007.

- [43] S. P. Poplack, T. D. Tosteson, C. A. Kogel, and H. M. Nagy, "Digital breast tomosynthesis: initial experience in 98 women with abnormal digital screening mammography", *AJR. American Journal of Roentgenology*, vol. 189, no. 3, pp. 616–623, 2007.
- [44] *Medical Devices, Selenia Dimensions 3D System – P080003*, U.S. Food and Drug Administration, 2011. [Online]. Available: <http://www.fda.gov/MedicalDevices/ProductsandMedicalProcedures/DeviceApprovalsandClearances/Recently-ApprovedDevices/ucm246400.htm> (visited on Feb. 22, 2014).
- [45] E. A. Rafferty, "Digital mammography: novel applications", *Radiologic Clinics of North America*, vol. 45, no. 5, pp. 831–843, 2007, Breast Imaging.
- [46] W. F. Good, G. S. Abrams, V. J. Catullo, D. M. Chough, M. A. Ganott, C. M. Hakim, and D. Gur, "Digital breast tomosynthesis: a pilot observer study", *AJR. American Journal of Roentgenology*, vol. 190, no. 4, pp. 865–869, 2008.
- [47] I. Andersson, D. Ikeda, S. Zackrisson, M. Ruschin, T. Svahn, P. Timberg, and A. Tingberg, "Breast tomosynthesis and digital mammography: a comparison of breast cancer visibility and birads classification in a population of cancers with subtle mammographic findings", *European Radiology*, vol. 18, no. 12, pp. 2817–2825, 2008.
- [48] S. Vandenberghe, Y. D'Asseler, R. V. de Walle, T. Kauppinen, M. Koole, L. Bouwens, K. V. Laere, I. Lemahieu, and R. A. Dierckx, "Iterative reconstruction algorithms in nuclear medicine", *Computerized Medical Imaging and Graphics*, vol. 25, no. 2, pp. 105–111, 2001.
- [49] Y. Zhang, H.-P. Chan, B. Sahiner, J. Wei, M. M. Goodsitt, L. M. Hadjiiski, J. Ge, and C. Zhou, "A comparative study of limited-angle cone-beam reconstruction methods for breast tomosynthesis", *Medical Physics*, vol. 33, no. 10, pp. 3781–3795, 2006.
- [50] E. Y. Sidky, X. Pan, I. S. Reiser, R. M. Nishikawa, R. H. Moore, and D. B. Kopans, "Enhanced imaging of microcalcifications in digital breast tomosynthesis through improved image-reconstruction algorithms", *Medical Physics*, vol. 36, no. 11, pp. 4920–4932, 2009.
- [51] T. Wu, R. H. Moore, E. A. Rafferty, and D. B. Kopans, "A comparison of reconstruction algorithms for breast tomosynthesis", *Medical Physics*, vol. 31, no. 9, pp. 2636–2647, 2004.
- [52] T. Gomi, "A comparison of reconstruction algorithms regarding exposure dose reductions during digital breast tomosynthesis", *Journal of Biomedical Science and Engineering*, vol. 7, no. 8, pp. 516–525, 2014.
- [53] A. K. Hara, R. G. Paden, A. C. Silva, J. L. Kujak, H. J. Lawder, and W. Pavlicek, "Iterative reconstruction technique for reducing body radiation dose at CT: feasibility study", *American Journal of Roentgenology*, vol. 193, no. 3, pp. 764–771, 2009.

- [54] D. Bliss, *Breast Anatomy*, National Cancer Institute. [Online]. Available: <https://visualsonline.cancer.gov/details.cfm?imageid=9306> (visited on Aug. 2, 2014).
- [55] A. Adam, A. K. Dixon, R. G. Grainger, and D. J. Allison, *Grainger & Allison's Diagnostic Radiology: A Textbook of Medical Imaging*. Churchill Livingstone, 2008.
- [56] R. R. Seeley, P. Tate, and T. D. Stephens, *Anatomy and Physiology*. McGraw-Hill Companies, 2008.
- [57] V. Harmer, *Breast Cancer Nursing Care and Management*. Wiley, 2011.
- [58] D. Carter, S. J. Schnitt, and R. R. Millis, "Sternberg's diagnostic surgical pathology", in S. E. Mills, Ed. 2012, ch. The Breast.
- [59] P. B. Gordon and S. L. Goldenberg, "Malignant breast masses detected only by ultrasound. A retrospective review", *Cancer*, vol. 76, no. 4, pp. 626–630, 1995.
- [60] A. Thomas, D. Thickman, C. L. Rapp, M. A. Dennis, S. H. Parker, and G. Sisney, "Solid breast nodules: use of sonography to distinguish between benign and malignant lesions", *Radiology*, vol. 196, pp. 123–134, 1995.
- [61] E. A. Morris, "Review of breast MRI: indications and limitations", *Seminars in Roentgenology*, vol. 36, no. 3, pp. 226–237, 2001.
- [62] E. Barkova and S. Burrell, "Nuclear medicine in the imaging and management of breast cancer", *Contemporary Diagnostic Radiology*, vol. 34, no. 26, pp. 1–5, 2011.
- [63] R. Patel, A. Khan, D. Wirth, M. Kamionek, D. Kandil, R. Quinlan, and A. N. Yaroslavsky, "Multimodal optical imaging for detecting breast cancer", *Journal of Biomedical Optics*, vol. 17, no. 6, pp. 0 660 081–0 660 089, 2012.
- [64] S. M. W. Y. Van de Ven, S. G. Elias, C. T. Chan, Z. Miao, Z. Cheng, A. De, and S. S. Gambhir, "Optical imaging with Her2-targeted affibody molecules can monitor Hsp90 treatment response in a breast cancer xenograft mouse model", *Clinical Cancer Research*, vol. 18, no. 4, pp. 1073–1081, 2012.
- [65] V. Kalles, G. C. Zografos, X. Provatopoulou, D. Koulocheri, and A. Gounaris, "The current status of positron emission mammography in breast cancer diagnosis", *Breast Cancer*, vol. 20, no. 2, pp. 123–130, 2013.
- [66] K. Schilling, D. Narayanan, J. Kalinyak, J. The, M. Velasquez, S. Kahn, M. Saady, R. Mahal, and L. Chrystal, "Positron emission mammography in breast cancer presurgical planning: comparisons with magnetic resonance imaging", *European Journal of Nuclear Medicine and Molecular Imaging*, vol. 38, no. 1, pp. 23–36, 2011.

- [67] T. Yanagida, A. Yoshikawa, Y. Yokota, K. Kamada, Y. Usuki, S. Yamamoto, M. Miyake, M. Baba, K. Kumagai, K. Sasaki, M. Ito, N. Abe, Y. Fujimoto, S. Maeo, Y. Furuya, H. Tanaka, A. Fukabori, T. Rodrigues dos Santos, M. Takeda, and N. Ohuchi, "Development of Pr:LuAG scintillator array and assembly for positron emission mammography", *Nuclear Science, IEEE Transactions on*, vol. 57, no. 3, pp. 1492–1495, 2010.
- [68] A. H. Golnabi, P. M. Meaney, S. Geimer, and K. D. Paulsen, "Microwave imaging for breast cancer detection and therapy monitoring", in *Biomedical Wireless Technologies, Networks, and Sensing Systems (BioWireless)*, 2011 IEEE Topical Conference on, 2011, pp. 59–62.
- [69] S. M. Aguilar, M. A. Al-Joumayly, J. D. Shea, N. Behdad, and S. C. Hagness, "Design of a microwave breast imaging array composed of dual-band miniaturized antennas", in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, 2011, pp. 1–4.
- [70] J. C. Y. Lai, C. B. Soh, E. Gunawan, and K. S. Low, "UWB microwave imaging for breast cancer detection — experiments with heterogeneous breast phantoms", *Progress In Electromagnetics Research M*, vol. 16, pp. 19–29, 2011.
- [71] I. Reiser and S. Glick, *Tomosynthesis Imaging*, ser. Imaging in medical diagnosis and therapy. Taylor & Francis, 2014.
- [72] A. Smith, *Design considerations in optimizing a breast tomosynthesis system*, Hologic Inc.
- [73] J. T. Dobbins III, "Tomosynthesis imaging: at a translational crossroads", *Medical Physics*, vol. 36, no. 6, pp. 1956–1967, 2009.
- [74] J. A. Baker and J. Y. Lo, "Breast tomosynthesis: state-of-the-art and review of the literature", *Academic Radiology*, vol. 18, no. 10, pp. 1298–1310, 2011.
- [75] R. Holland, M. Mravunac, J. H. C. L. Hendriks, and B. V. Bekker, "So-called interval cancers of the breast: pathologic and radiologic analysis of sixty-four cases", *Cancer*, vol. 49, no. 12, pp. 2527–2533, 1982.
- [76] J. E. Martin, M. Moskowitz, and J. R. Milbrath, "Breast cancer missed by mammography", *AJR. American Journal of Roentgenology*, vol. 132, no. 5, pp. 737–739, 1979.
- [77] R. E. Bird, T. W. Wallace, and B. C. Yankaskas, "Analysis of cancers missed at screening mammography", *Radiology*, vol. 184, no. 3, pp. 613–617, 1992.
- [78] E. A. Rafferty, "Digital mammography: novel applications", *Radiologic Clinics of North America*, vol. 45, no. 5, pp. 831–843, 2007.
- [79] M. K. Barton, "Digital breast tomosynthesis may improve breast cancer detection rates", *CA: A Cancer Journal for Clinicians*, vol. 63, no. 5, pp. 291–292, 2013.

- [80] D. Bernardi, F. Caumo, P. Macaskill, S. Ciatto, M. Pellegrini, S. Brunelli, P. Tuttonbene, P. Bricolo, C. Fantò, M. Valentini, S. Montemezzi, and N. Houssami, "Effect of integrating 3D-mammography (digital breast tomosynthesis) with 2D-mammography on radiologists' true-positive and false-positive detection in a population breast screening trial", *European Journal of Cancer*, pp. 1–7, 2014.
- [81] T. Uematsu, "The emerging role of breast tomosynthesis", *Breast Cancer*, vol. 20, no. 3, pp. 204–212, 2013.
- [82] D. Bernardi, S. Ciatto, M. Pellegrini, V. Anesi, S. Burlon, E. Cauli, M. Depaoli, L. Larentis, V. Malesani, L. Targa, P. Baldo, and N. Houssami, "Application of breast tomosynthesis in screening: incremental effect on mammography acquisition and reading time", *The British Journal of Radiology*, vol. 85, no. 1020, e1174–e1178, 2012.
- [83] D. Förnvik, I. Andersson, T. Svahn, P. Timberg, S. Zackrisson, and A. Tingberg, "The effect of reduced breast compression in breast tomosynthesis: human observer study using clinical cases", *Radiation Protection Dosimetry*, vol. 139, no. 1-3, pp. 118–123, 2010.
- [84] R. S. Saunders Jr, E. Samei, J. Y. Lo, and J. A. Baker, "Can compression be reduced for breast tomosynthesis? Monte carlo study on mass and microcalcification conspicuity in tomosynthesis", *Radiology*, vol. 251, no. 3, pp. 673–682, 2009.
- [85] T. Svahn, K. Lång, I. Andersson, and S. Zackrisson, "Differences in radiologists' experiences and performance in breast tomosynthesis", in *Breast Imaging*, Springer, 2012, pp. 377–385.
- [86] D. D. Dershaw, "Large core needle biopsy with tomosynthesis guidance: another development in breast imaging technology", *The Breast Journal*, vol. 19, no. 1, pp. 1–3, 2013.
- [87] L. Vancamberg, A. Sahbani, S. Muller, and G. Morel, "Needle path planning for digital breast tomosynthesis biopsy using a heterogeneous model", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 5749–5755.
- [88] J. Viala, P. Gignier, B. Perret, C. Hovasse, D. Hovasse, M.-D. Chancelier-Galan, G. Borner, A. Hamrouni, J.-l. Lasry, and J.-P. Convard, "Stereotactic vacuum-assisted biopsies on a digital breast 3D-tomosynthesis system", *The Breast Journal*, vol. 19, no. 1, pp. 4–9, 2013.
- [89] A. Smith, *Fundamentals of breast tomosynthesis, Improving the performance of mammography*, Hologic Inc., 2008.
- [90] J. T. Dobbins III and D. J. Godfrey, "Digital X-ray tomosynthesis: current state of the art and clinical potential", *Physics in Medicine and Biology*, vol. 48, no. 19, 2003.
- [91] A. Smith, "Full-field breast tomosynthesis", *Radiology management*, vol. 27, no. 5, 2005.

- [92] D. G. Grant, "Tomosynthesis: a three-dimensional radiographic imaging technique", *Biomedical Engineering, IEEE Transactions on*, no. 1, pp. 20–28, 1972.
- [93] L. T. Niklason, B. T. Christian, L. E. Niklason, *et al.*, "Digital tomosynthesis in breast imaging", *Radiology*, vol. 205, no. 2, pp. 399–406, 1997.
- [94] G. L. Zeng, "Image reconstruction — a tutorial", *Computerized Medical Imaging and Graphics*, vol. 25, no. 2, pp. 97–103, 2001.
- [95] G. Lauritsch and W. H. Härer, "Theoretical framework for filtered back projection in tomosynthesis", in *Medical Imaging'98*, International Society for Optics and Photonics, 1998, pp. 1127–1137.
- [96] B. E. H. Claus, J. W. Eberhard, A. Schmitz, P. Carson, M. Goodsitt, and H.-P. Chan, "Generalized filtered back-projection reconstruction in breast tomosynthesis", in *Digital Mammography*, ser. Lecture Notes in Computer Science, S. M. Astley, M. Brady, C. Rose, and R. Zwigelaar, Eds., vol. 4046, Springer Berlin Heidelberg, 2006, pp. 167–174.
- [97] A. Teymurazyan, T. Riauka, H.-S. Jans, and D. Robinson, "Properties of noise in positron emission tomography images reconstructed with filtered-backprojection and row-action maximum likelihood algorithm", *Journal of Digital Imaging*, vol. 26, no. 3, pp. 447–456, 2013.
- [98] G. T. Gullberg and G. L. Zeng, "A cone-beam filtered backprojection reconstruction algorithm for cardiac single photon emission computed tomography", *Medical Imaging, IEEE Transactions on*, vol. 11, no. 1, pp. 91–101, 1992.
- [99] C. X. Wang, W. E. Snyder, G. Bilbro, and P. Santago, "Performance evaluation of filtered backprojection reconstruction and iterative reconstruction methods for PET images", *Computers in Biology and Medicine*, vol. 28, no. 1, pp. 13–25, 1998.
- [100] J. Li, R. J. Jaszczak, K. L. Greer, and R. E. Coleman, "A filtered backprojection algorithm for pinhole SPECT with a displaced centre of rotation", *Physics in Medicine and Biology*, vol. 39, no. 1, 1994.
- [101] N.-Y. Lee and Y. Choi, "A modified OSEM algorithm for PET reconstruction using wavelet processing", *Computer Methods and Programs in Biomedicine*, vol. 80, no. 3, pp. 236–245, 2005.
- [102] B. Li, G. B. Avinash, J. W. Eberhard, and B. E. H. Claus, "Optimization of slice sensitivity profile for radiographic tomosynthesis", *Medical Physics*, vol. 34, pp. 2907–2916, 2007.
- [103] R. Gordon, R. Bender, and G. T. Herman, "Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography", *Journal of Theoretical Biology*, vol. 29, no. 3, pp. 471–481, 1970.
- [104] P. Gilbert, "Iterative methods for the three-dimensional reconstruction of an object from projections", *Journal of Theoretical Biology*, vol. 36, no. 1, pp. 105–117, 1972.

- [105] A. H. Andersen and A. C. Kak, "Simultaneous algebraic reconstruction technique (SART): a superior implementation of the ART algorithm", *Ultrasonic Imaging*, vol. 6, no. 1, pp. 81–94, 1984.
- [106] M. N. Wernick and J. N. Aarsvold, *Emission Tomography: The Fundamentals of PET and SPECT*. Elsevier Science, 2004.
- [107] K. Lange, R. Carson, *et al.*, "EM reconstruction algorithms for emission and transmission tomography", *Journal of Computer Assisted Tomography*, vol. 8, no. 2, pp. 306–316, 1984.
- [108] G. L. Zeng, *Medical Image Reconstruction: A Conceptual Tutorial*. Higher Education Press, 2010.
- [109] J. A. Browne and T. J. Holmes, "Developments with maximum-likelihood X-ray computed tomography: initial testing with real data", *Applied Optics*, vol. 33, no. 14, pp. 3010–3022, 1994.
- [110] H. Hudson and R. Larkin, "Accelerated image reconstruction using ordered subsets of projection data", *Medical Imaging, IEEE Transactions on*, vol. 13, no. 4, pp. 601–609, 1994.
- [111] C. A. Mack, "Fifty years of Moore's law", *Semiconductor Manufacturing, IEEE Transactions on*, vol. 24, no. 2, pp. 202–207, 2011.
- [112] G. Shen, G.-P. Gao, S. Li, H.-Y. Shum, and Y.-Q. Zhang, "Accelerate video decoding with generic GPU", *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 5, pp. 685–693, 2005.
- [113] *Growth of GPU Computing*, NVIDIA, 2014. [Online]. Available: <http://nvidiaianews.nvidia.com/> (visited on Mar. 7, 2014).
- [114] L.-C. Chang, E. El-Araby, V. Q. Dang, and L. H. Dao, "GPU acceleration of non-linear diffusion tensor estimation using CUDA and MPI", *Neurocomputing*, 2014.
- [115] E. Solomou, S. Kostopoulos, K. Sidiropoulos, E. Athanasiadis, E. Lavdas, D. Glotsos, G. Sakellaropoulos, P. Zampakis, J. Stonham, and D. Cavouras, "Designing a pattern recognition system on GPU for discriminating between patients with micro-ischaemic and multiple sclerosis lesions, using MRI images", *International Journal of High Performance Computing Applications*, vol. 27, no. 3, pp. 348–359, 2013.
- [116] T. Idzenga, E. Gaburov, W. Vermin, J. Menssen, and C. De Korte, "Fast 2-D ultrasound strain imaging: the benefits of using a GPU", *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 61, no. 1, pp. 207–213, 2014.
- [117] J. P. Asen, J. I. Buskenes, C.-I. C. Nilsen, A. Austeng, and S. Holm, "Implementing capon beamforming on a GPU for real-time cardiac ultrasound imaging", *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, vol. 61, no. 1, pp. 76–85, 2014.

- [118] L. A. Flores, V. Vidal, P. Mayo, F. Rodenas, and G. Verdú, "Parallel CT image reconstruction based on GPUs", *Radiation Physics and Chemistry*, vol. 95, pp. 247–250, 2014.
- [119] P. B. Noël, A. M. Walczak, J. Xu, J. J. Corso, K. R. Hoffmann, and S. Schafer, "GPU-based cone beam computed tomography", *Computer Methods and Programs in Biomedicine*, vol. 98, no. 3, pp. 271–277, 2010.
- [120] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities", in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring), Atlantic City, New Jersey: ACM, 1967, pp. 483–485.
- [121] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, 2nd edition. Elsevier Science, 2012.
- [122] OpenCL, Khronos Group. [Online]. Available: <https://www.khronos.org/opencl/> (visited on Sep. 11, 2014).
- [123] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Pearson Education, 2010.
- [124] M. Harvey and G. D. Fabritiis, "Swan: a tool for porting CUDA programs to OpenCL", *Computer Physics Communications*, vol. 182, no. 4, pp. 1093–1099, 2011.
- [125] W. W. Hwu, *GPU Computing Gems Emerald Edition*. Elsevier, 2011.
- [126] D. Vintache, B. Humbert, and D. Brasse, "Iterative reconstruction for transmission tomography on GPU using nvidia CUDA", *Tsinghua Science & Technology*, vol. 15, no. 1, pp. 11–16, 2010.
- [127] J. S. Kole and F. J. Beekman, "Evaluation of accelerated iterative X-ray CT image reconstruction using floating point graphics hardware", *Physics in Medicine and Biology*, vol. 51, no. 4, 2006.
- [128] X. Jia, Y. Lou, J. Lewis, R. Li, X. Gu, C. Men, W. Y. Song, and S. B. Jiang, "GPU-based fast low-dose cone beam CT reconstruction via total variation", *Journal of X-Ray Science and Technology*, vol. 19, no. 2, pp. 139–154, 2011.
- [129] X. Jia, Y. Lou, R. Li, W. Y. Song, and S. B. Jiang, "GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation", *Medical Physics*, vol. 37, no. 4, pp. 1757–1760, 2010.
- [130] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware", *Nuclear Science, IEEE Transactions on*, vol. 52, no. 3, pp. 654–663, 2005.
- [131] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, and J. Hornegger, "GPU-accelerated SART reconstruction using the CUDA programming environment", in *SPIE Medical Imaging*, International Society for Optics and Photonics, 2009.

- [132] I. Goddard, T. Wu, S. Thieret, A. Berman, and H. Bartsch, "Implementing an iterative reconstruction algorithm for digital breast tomosynthesis on graphics processing hardware", in *Proc. SPIE*, vol. 6142, 2006, pp. 61424V–61424V–7.
- [133] D. Schaa, B. Brown, B. Jang, P. Mistry, R. Dominguez, D. Kaeli, R. Moore, and D. B. Kopans, "Chapter 40 - GPU acceleration of iterative digital breast tomosynthesis", in *GPU Computing Gems Emerald Edition*, ser. Applications of GPU Computing Series, W.-m. W. Hwu, Ed., Boston: Morgan Kaufmann, 2011, pp. 647–657.
- [134] G. Pratz, G. Chinn, P. D. Olcott, and C. S. Levin, "Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU", *Medical Imaging, IEEE Transactions on*, vol. 28, no. 3, pp. 435–445, 2009.
- [135] Siemens MAMMOMAT Inspiration. [Online]. Available: <http://usa.healthcare.siemens.com/mammography/digital-mammography/mammomat-inspiration> (visited on Aug. 3, 2014).
- [136] NHSBSP Equipment Report 1306, *Technical evaluation of siemens mammomat inspiration digital breast tomosynthesis system*, NHS Cancer Screening Programme, Oct. 2013. [Online]. Available: <http://www.cancerscreening.nhs.uk/breastscreen/publications/mammography-equipment.html> (visited on Aug. 8, 2014).
- [137] T. Mertelmeier, J. Speitel, and C. Frumento, *3D breast tomosynthesis—intelligent technology for clear clinical benefits*, Siemens AG.
- [138] Q. Li, Y. Wang, H. Liu, X. He, D. Xu, J. Wang, and F. Guo, "Leukocyte cells identification and quantitative morphometry based on molecular hyperspectral imaging technology", *Computerized Medical Imaging and Graphics*, vol. 38, no. 3, pp. 171–178, 2014.
- [139] A. Moignier, D. Broggio, S. Derreumaux, F. E. Baf, A.-M. Mandin, T. Girinsky, J.-F. Paul, M. Chea, C. Jenny, D. Franck, B. Aubert, and J.-J. Mazon, "Dependence of coronary 3-dimensional dose maps on coronary topologies and beam set in breast radiation therapy: a study based on CT angiographies", *International Journal of Radiation Oncology*Biology*Physics*, vol. 89, no. 1, pp. 182–190, 2014.
- [140] W. Schweitzer, T. D. Ruder, M. J. Thali, and H. Ringl, "1.11. Skull fractures in post mortem CT: VRT, flat and skin surface projections in comparison", *Journal of Forensic Radiology and Imaging*, vol. 2, no. 2, p. 98, 2014.
- [141] J. Zhou, E. Tryggstad, Z. Wen, B. Lal, T. Zhou, R. Grossman, S. Wang, K. Yan, D.-X. Fu, E. Ford, *et al.*, "Differentiation between glioma and radiation necrosis using molecular magnetic resonance imaging of endogenous proteins and peptides", *Nature Medicine*, vol. 17, no. 1, pp. 130–134, 2011.

- [142] D. Messroghli, A. Rudolph, H. Abdel-Aty, R. Wassmuth, T. Kuhne, R. Dietz, and J. Schulz-Menger, "An open-source software tool for the generation of relaxation time maps in magnetic resonance imaging", *BMC Medical Imaging*, vol. 10, no. 1, 2010.
- [143] K. P. Bowman, *An Introduction to Programming with IDL: Interactive Data Language*. Elsevier Science, 2006.
- [144] *GPULib*, Tech-X Corporation. [Online]. Available: <http://www.txcorp.com/home/gpulib> (visited on Aug. 19, 2014).
- [145] R. L. Siddon, "Fast calculation of the exact radiological path for a three-dimensional CT array", *Medical Physics*, vol. 12, no. 2, pp. 252–255, 1985.
- [146] F. Jacobs, E. Sundermann, B. De Sutter, M. Christiaens, and I. Lemahieu, "A fast algorithm to calculate the exact radiological path through a pixel or voxel space", *CIT. Journal of Computing and Information Technology*, vol. 6, no. 1, pp. 89–94, 1998.
- [147] *GPU-Accelerated Libraries*, NVIDIA. [Online]. Available: <https://developer.nvidia.com/gpu-accelerated-libraries> (visited on Aug. 16, 2014).
- [148] N. M. Josuttis, *The C++ Standard Library: A Tutorial and Reference*. Pearson Education, 2012.
- [149] D. Sharma, A. Badal, and A. Badano, "hybridMANTIS: a CPU-GPU monte carlo method for modeling indirect X-ray detectors with columnar scintillators", *Physics in Medicine and Biology*, vol. 57, no. 8, pp. 2357–2372, 2012.
- [150] *CUDA GPUs*, NVIDIA. [Online]. Available: <https://developer.nvidia.com/cuda-gpus> (visited on Sep. 2, 2014).
- [151] *Quadro FX 3800*, NVIDIA. [Online]. Available: http://www.nvidia.com/object/product_quadro_fx_3800_us.html (visited on Aug. 20, 2014).
- [152] *Quadro 600*, NVIDIA. [Online]. Available: <http://www.nvidia.com/object/product-quadro-600-us.html> (visited on Aug. 20, 2014).
- [153] *CALL_EXTERNAL*, EXELIS Visual Information Solutions. [Online]. Available: http://www.exelisvis.com/docs/CALL_EXTERNAL.html (visited on Aug. 20, 2014).
- [154] *LINKIMAGE*, EXELIS Visual Information Solutions. [Online]. Available: <http://www.exelisvis.com/docs/LINKIMAGE.html> (visited on Aug. 20, 2014).
- [155] R. Kling, *Calling C from IDL: Using DLM's to Extend Your IDL Code*. Kling Research and Software, 2003.
- [156] N. Oliveira, N. Matela, R. Bugalho, N. Ferreira, and P. Almeida, "Optimization of 2D image reconstruction for positron emission mammography using IDL", *Computers in Biology and Medicine*, vol. 39, no. 2, pp. 119–129, 2009.

- [157] *GeForce GTX 780*, NVIDIA GeForce. [Online]. Available: <http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-780> (visited on Sep. 10, 2014).
- [158] *Unified Memory in CUDA 6*, NVIDIA. [Online]. Available: <http://devblogs.nvidia.com/paralleforall/unified-memory-in-cuda-6/> (visited on Sep. 9, 2014).
- [159] *AMD Graphics*, AMD. [Online]. Available: <http://www.amd.com/en-us/products/graphics> (visited on Sep. 11, 2014).
- [160] *Intel Xeon Phi Product Family*, Intel Corporation. [Online]. Available: <http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html> (visited on Sep. 11, 2014).
- [161] *OpenMP*. [Online]. Available: <http://openmp.org/wp/> (visited on Sep. 11, 2014).
- [162] *OpenACC*. [Online]. Available: <http://www.openacc-standard.org/> (visited on Sep. 11, 2014).



Poster of WBME

This appendix contains the poster presented in the 6th Workshop on Biomedical Engineering (WBME), held in Lisbon, Portugal, on the 5th April 2014. Here, some initial results obtained in this dissertation with a single GPU-implementation were shown.

Abstract

In this work a parallelized iterative algorithm (OS-EM) was implemented, using CUDA™ to program GPUs, for Breast Tomosynthesis Image Reconstruction. The results showed that the GPU implementation allowed a decrease in the reconstruction times of approximately 4.6 times, compared with the CPU implementation.

Introduction

- All over the world, X-ray mammography is the current gold-standard for medical imaging of breast cancer [1]. However, it is associated with some well-known limitations which can be overcome by digital breast tomosynthesis (DBT).
- DBT is a 3D radiographic technique that reduces the obscuring effect of tissue overlap (Figure 1) and appears to address both issues of false-negative and false-positive rates [3, 4].
- The 3D images in DBT are only achieved through reconstruction methods. In a clinical setting, these methods need to be both accurate and fast.
- Two major groups of reconstruction algorithms are:
 - Analytical** – the most used nowadays, although employing analytical models that simplify reconstructions;
 - Iterative** – uses geometric models with much more precision, producing higher-quality images, but are much more computationally intensive.
- Generally, every iterative image reconstruction algorithm can be explained using the scheme presented in Figure 2.
- The performance optimization of iterative algorithms can be achieved via parallel computing by being implemented in platforms such as graphics processing units (GPUs) to make the 3D reconstruction faster.

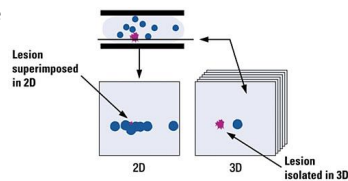


Figure 1. Tissue overlap in X-ray mammography (2D). With DBT (3D), the cross-sectional slices make the lesion less likely to be obscured [2].



Figure 3. Siemens MAMMOMAT Inspiration.

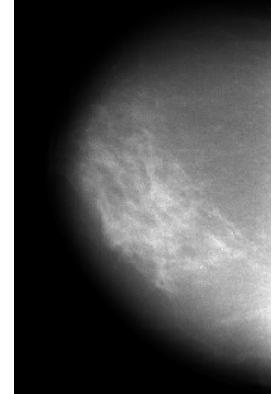


Figure 4. Image obtained from the GPU implementation of OS-EM [5].

Results

- In Figure 4 it is possible to observe a section of the obtained image with the GPU implementation.
- The reconstruction times were measured by the OS-EM algorithm with 25 subsets (i.e., the *Update* shown in Figure 1 is done for each projection) and preliminary results are shown in Table 1.

Table 1. Preliminary results of image reconstruction times on GPU and CPU and the obtained speedup.

	Reconstruction time (seconds)	Speedup ($\frac{t_{CPU}}{t_{GPU}}$)
GPU	1011	4.64 x
CPU	4692	

Discussion

- The preliminary results are promising from a clinical point of view – reduction in the reconstruction time from approximately 80 to 15 minutes.
- Improvements in the code are possible to be accomplished, which may lead to even more impressive results.
- The only current limitation are the memory factors – with a GPU with more memory, it would be possible to achieve a higher speedup.

Conclusions

- Significant reduction in reconstruction time was achieved with GPU (approximately 4.6 times).
- GPU proved to be a very effective device for numeric processing.
- The obtained time reduction in the image reconstruction may increase the popularity of iterative algorithms for image reconstruction, in practice.
- As future work, I propose to:
 - Improve functions programmed in CUDA;
 - Study the code performance in a GPU with higher memory capacity;
 - Quantitatively evaluate the image quality;
 - Parallelize other reconstruction algorithms.

References

- M. L. Brown, F. Houn, E. A. Sickles, and L. G. Kessler. "Screening Mammography in Community Practice: Positive Predictive Value of Abnormal Findings and Yield of Follow-up Diagnostic Procedures". In: *AJR, American Journal of Roentgenology* 165.6 (1995), pp. 1373–1377.
- A. Smith. Design Considerations in Optimizing a Breast Tomosynthesis System. Available from <http://www.hologic.com/en/learning-center/white-papers/breastimaging/> [accessed on 2/04/2014]. Hologic Inc.
- J. M. Park, E. A. Franken, M. Gang, L. L. Fajardo, and L. T. Niklason. "Breast Tomosynthesis: Present Considerations and Future Applications". In: *Radiographics* 27 Suppl 1 (2007), S231–S240.
- S. P. Poplack, T. D. Tosteson, C. A. Kogel, and H. M. Nagy. "Digital Breast Tomosynthesis: Initial Experience in 98 Women with Abnormal Digital Screening Mammography". In: *AJR, American Journal of Roentgenology* 189.3 (2007), pp. 616–623.
- B. Azevedo. "Reconstrução/processamento de imagem médica com GPU em tomosíntese." (2011).

Methods and Materials

- The iterative algorithm that was chosen to be accelerated was the Ordered Subsets – Expectation Maximization (OS-EM):
 - good results shown;
 - CPU implementation already available.
- The exam used in this work was performed by a DBT system from the Hospital da Luz, Portugal. The DBT system is the Siemens MAMMOMAT Inspiration (Figure 3), which performs 25 projections in each exam in an angular range of -25° to $+25^\circ$.
- Both the CPU implementation and the parallelized were performed in the same CPU. The GPU used was an NVIDIA Quadro 600 with 1 GB of memory and 96 CUDA cores.
- The 5.5 version of the CUDA Toolkit was the one used, taking advantage of some functions of the Thrust library.
- The final image resolution was changed using a scaling factor of 4 due to memory constraints.

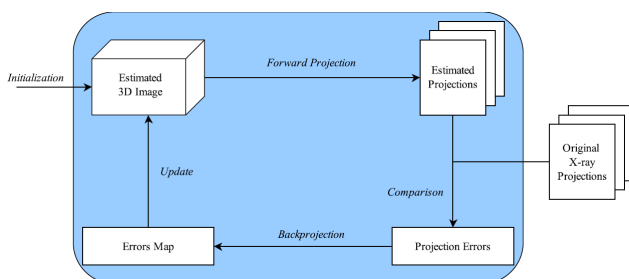


Figure 2. Schematic of iterative image reconstruction algorithm for digital breast tomosynthesis.



Poster of PUMPS

This appendix contains the poster presented in the 5th edition of the Programming and Tuning Massively Parallel Systems summer school (PUMPS), held in Barcelona, Spain, from the 7th to 11th July 2014. Here, preliminary results with the CUDA-IDL implementation were provided.

Breast Tomosynthesis Image Reconstruction using GPGPUs

Pedro Ferreira^{1,2}, Nuno Oliveira², Nuno Matela² and Pedro Medeiros¹

¹Faculdade de Ciências e Tecnologia, Universidade NOVA de Lisboa, Lisbon, Portugal

²Instituto de Biofísica e Engenharia Biomédica, Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal

Abstract

In this work an IDL™ program was partially parallelized for iterative algorithms, using CUDA™ to program GPUs, for Breast Tomosynthesis Image Reconstruction. Preliminary results showed that this implementation allowed a decrease in the reconstruction times of approximately 1.5 times, compared with the CPU implementation.

Introduction

- **Breast cancer** is the most common cancer among women.
- X-ray mammography is the current gold-standard for medical imaging of breast cancer [1], however it has some well-known limitations, which prompt the development of **digital breast tomosynthesis (DBT)**, a 3D radiographic technique (Figure 1).

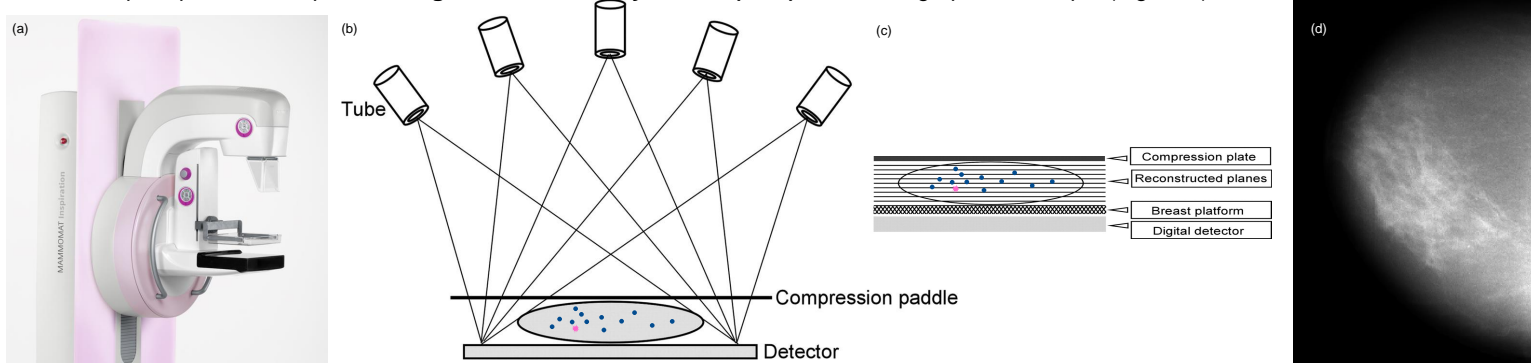


Figure 1. Basic principles of breast tomosynthesis. (a) Siemens MAMMOMAT Inspiration. (b) Image data are acquired from various angles as the X-ray tube moves. (c) The cross-sectional slices make the pathologies (pink lesion) less likely to be obscured (in opposite to the overlap that would appear in X-ray mammography – 2D). (d) Single reconstructed plane from the 3D image.

- The aim of this project is to develop a **fast DBT-based image reconstruction** method that includes the implementation of 3 iterative algorithms (ART, ML-EM and OS-EM) on **graphics processing units (GPUs)** using **CUDA™**.

Methods

- The medical exam used was performed by a DBT system from the Hospital da Luz, Portugal.
- A sequential implementation made in IDL™ (Interactive Data Language) is the basis of our work.
- Previous work [2] has proved that the use of CUDA™ will enhance significantly the execution times of the reconstruction process, so we start by allowing the IDL™ program to call CUDA™ code - Figure 2.

6.2 speedup $\left(\frac{t_{CPU}}{t_{GPU}}\right)$ with a GPU-implementation of OSEM

- A recent version of the NVIDIA® CUDA® Toolkit is used, taking advantage of some functions of the Thrust library.

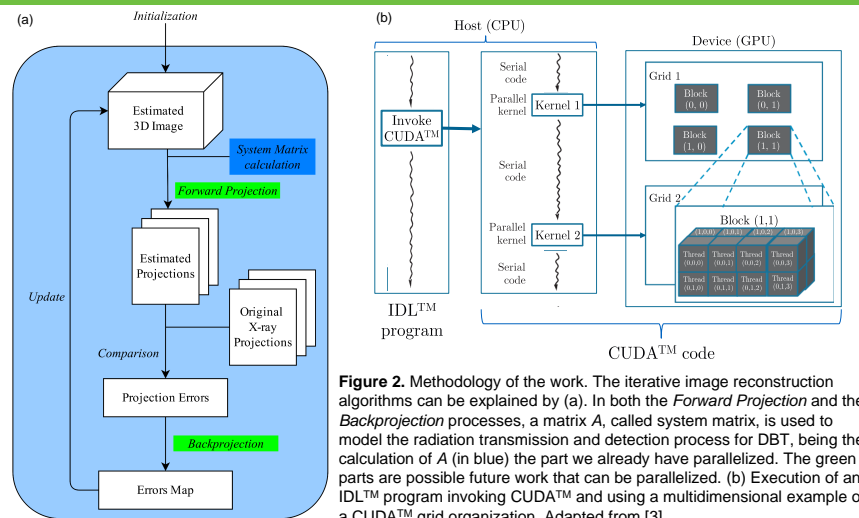


Figure 2. Methodology of the work. The iterative image reconstruction algorithms can be explained by (a). In both the *Forward Projection* and the *Backprojection* processes, a matrix *A*, called system matrix, is used to model the radiation transmission and detection process for DBT, being the calculation of *A* (in blue) the part we already have parallelized. The green parts are possible future work that can be parallelized. (b) Execution of an IDL™ program invoking CUDA™ and using a multidimensional example of a CUDA™ grid organization. Adapted from [3].

Preliminary Results and Conclusions

- For now, CUDA™ code provides the calculation of the system matrix (Figure 2), which is a computationally expensive part of the reconstruction algorithm. The parallelization of other parts is on-going work; the integration between IDL™ and CUDA™ allows an incremental development of the image reconstruction program, by replacing sequential modules in IDL™ by parallelized ones as soon as they are available.
- With this approach we expect to:
 1. develop an **accurate** and a **fast** process with turnaround times compatible with its use in a **real clinical setting**;
 2. **reduce the radiation dose** received by patients during a medical examination.
- This process may be adapted to other medical techniques such as computed tomography and magnetic resonance imaging.

Preliminary results
with ART
1.5 speedup

References

1. M. L. Brown, F. Houn, E. A. Sickles, and L. G. Kessler. "Screening Mammography in Community Practice: Positive Predictive Value of Abnormal Findings and Yield of Follow-up Diagnostic Procedures". In: AJR. American Journal of Roentgenology 165.6 (1995), pp. 1373–1377.
2. B. Azevedo. "Reconstrução/processamento de imagem médica com GPU em tomossíntese." (2011).
3. D. B. Kirk and W. W. Hwu. Programming Massively Parallel Processors: A Hands-on Approach. 2nd edition. Elsevier Science, 2012.

Acknowledgements

This work was supported in part by Fundação para a Ciência e a Tecnologia - Portugal (grants numbers SFRH/BD/43376/2008, SFRH/BPD/29696/2006, SFRH/BD/47448/2008, SFRH/BD/66008/2009, SFRH/BD/33667/2009 and SFRH/BPD/64846/2009) and by Quadro de Referência Estratégico Nacional, QREN (Project PET II-b QREN 1590)





Quick Guide

This appendix contains the quick guide entitled "CUDA-IDL Implementation of Digital Breast Tomosynthesis Image Reconstruction". This guide is intended for users of the program developed in this dissertation and for future developers who will make use of this work.



IBEB

Instituto de Biofísica e Engenharia Biomédica

Quick Guide

CUDA-IDL Implementation of Digital Breast Tomosynthesis Image Reconstruction

for

Users and Future Developers

Pedro Rafael Tomé Ferreira

2014

Introduction

Tomosynthesis is an emerging radiological technique used for breast imaging which produces a 3D image of a series of views. There are two main groups of image reconstruction algorithms: iterative and analytical. This implementation deals with the iterative algorithms which are the ones that can improve image quality when compared to the others [1]. Here, in this **CUDA-IDL Implementation of Digital Breast Tomosynthesis Image Reconstruction**, three different algorithms are available:

- **SART** - Algebraic Reconstruction Technique [2];
- **ML-EM** – Maximum Likelihood – Expectation Maximization [3];
- **OS-EM** – Ordered Subsets – Expectation Maximization [4].

All of these algorithms make use of the GPU (Graphics Processing Unit) for the calculation of the system matrix (geometric model of the transmission and detection processes), which is the most computationally intensive part in the reconstruction process. The system matrix calculation is done using NVIDIA CUDA (Compute Unified Device Architecture) to program GPUs.

This guide is intended for users of this program and for future developers who will make use of this work.

System Requirements

System requirements to use this program:

- CUDA enabled GPU [5];
- CUDA software [6];
- Thrust library (it is already included in the most recent CUDA toolkit versions) [7];
- IDL software;
- NVIDIA device drivers.

During Pedro Ferreira's dissertation, this program was tested in the following two different systems:

Table 1 – Systems used in Pedro Ferreira's dissertation.

	System 1	System 2
Operating System	CentOS 6.5	
CPU	Intel Xeon E5530 (4x 2.40 GHz)	Intel Xeon W3505 (2x 2.53 GHz)
Main Memory	11.6 GB	23.4 GB
GPU	Quadro FX 3800	Quadro 600
Device Drivers	331.62	

Table 2 – Devices used in Pedro Ferreira's dissertation.

	Quadro FX 3800	Quadro 600
CUDA Cores	192	96
Memory Bandwith (GB/s)	51.2	25.6
Memory (GB)	1	
Compute Capability	1.3	2.1
Microarchitecture	Tesla	Fermi
Maximum Power Consumption (W)	108	40

For more information concerning the devices' specifications refer to [8], [9]. During Pedro Ferreira's dissertation the 6.0 CUDA Toolkit version was used, in which the Thrust version 1.7.0 and a compiler for NVIDIA GPUs known as NVCC, which stands for NVIDIA's CUDA Compiler, are included. The IDL Version that was used was the 8.2.2 for System 1 and 8.0 for System 2. In order to visualize and study each reconstructed image QuasiManager was used, an image visualization and data analysis software developed within the group.

Users of the program

Execute the program

In order to use this program it is necessary to have in the same folder the following files:

- `export.h` – required IDL header file (typically the header file is found in the `idl "external"` directory)
- `dbtreconstruction.pro` – IDL code
- `system_matrix_calc.cu` – CUDA code

To run this program in the IBEB computers make sure you are part of the **ibeb_mi** group by entering:

Id <your_username>

Then, it is necessary to compile the CUDA code. For that, the user should open the shell and go to the directory where the above files are and then execute the following command:

`nvcc -m64 -o libtest.so --shared -Xcompiler -fPIC system_matrix_calc.cu`

As soon as the CUDA file is compiled, a `"libtest.so"` file is created. Run IDL with the following command:

idl

In order to execute the program run the following instruction in IDL:

`linkimage, 'entry', './libtest.so', 1`

`.compile '/home/nmoliveira/QuasiManager/quasimakelog.pro'`

`.compile '/home/your_username/Desktop/testing/dbtreconstruction.pro'` ; directory where the `dbtreconstruction.pro` file is

`saveDir = '/home/your_username/Desktop/testing/'` ; fill with the directory where you want the reconstructions to be recorded

`fileName = 'myTest'` ; the names are combinations of this, such as: `myTest_parameter_parameter_iterationX`

`projDir` =
`'/partilha/MI/CAD_Mamografia/TomossÃ-ntese/Dados/CASO2__100728952/TOMOSCAN_UNILAT.DTO_20100503_185305_296000/TOMO_R-CC_TOMOSCAN_UNILAT.DTO_DIAGNOSIS_0003/'` ; directory where the projections to open are – if connected through SSH use this `projDir`, if directly in the PC use instead of `"TomossÃ-ntese"`, `"Tomossíntese"`

`pathToFiles = FILE_SEARCH(projDir, '*', COUNT = ct)`

The last instruction that is needed is the indication of which algorithm to use. Here, examples of the three different possibilities (SART, ML-EM and OS-EM) for a bin's scale factor of 16 and for a single iteration are shown:

- **`dbtreconstruction, pathToFiles, SAVEFILE = saveDir + fileName, SAVEDIR = saveDir, ALGORITHM = "ART", DOWNSCALE = 16, ITERATIONS = 1, RELAXATION = 0.2`** ; SART with a relaxation parameter of 0.2

- **dbtreconstruction, pathToFiles, SAVEFILE = saveDir + fileName, SAVEDIR = saveDir, ALGORITHM = "OSEM", DOWNSCALE = 16, ITERATIONS = 1, SUBSETS = 1 ; ML-EM**
- **dbtreconstruction, pathToFiles, SAVEFILE = saveDir + fileName, SAVEDIR = saveDir, ALGORITHM = "OSEM", DOWNSCALE = 16, ITERATIONS = 1, SUBSETS = 4 ; OS-EM with 4 subsets**

Note that ML-EM is a particular case of OS-EM (where the number of subsets is 1). In the end, the resulting images are generated with a log file with information about dimensions, type and date of reconstruction, which is useful to be possible to visualize the images through QuasiManager (QM).

Visualize reconstruction through QuasiManager

The generated images are binary files that can be opened by QM, an IDL based software created by Nuno Oliveira. To run the QM in the IBEB computers make sure you are part of the **ibeb_mi** group. Then copy the execution file **"quasimanager"** from Nuno Oliveira's folder to your home folder with the command:

```
cp /home/nmoliveira/QuasiManager/quasimanager ~/.
```

Finally run QM from your home directory by:

```
cd  
./quasimanager
```

On the left side panel, find the generated reconstructed images and open them. QM will automatically fill in the image dimensions according to the generated log file. In case the log file is missing, the default parameters are:

For a bin's scale factor of 16:

- Data type: **float**
- 1st (X) dimension: **176**
- 2nd (Y) dimension: **224**
- 3rd (Z) dimension: **60**

For a bin's scale factor of 4:

- Data type: **float**
- 1st (X) dimension: **704**
- 2nd (Y) dimension: **896**
- 3rd (Z) dimension: **25**

Further using of QM is beyond this user's guide.

Developers of the program

In this section several tips are provided for future developers of this program. For now, this program calculates the system matrix on the GPU, being the only part parallelized. The greatest advantages of the way this CUDA-IDL implementation is made are:

- The quality of results that can be assessed by comparison with pure-IDL implementation;
- A possible incremental approach, by replacing sequential parts of the computation by parallelized ones, as soon as available.

To find out your CUDA version, please use the command:

```
nvcc --version
```

It is possible to have different CUDA toolkit versions installed in the same system. In order to change the version that is being used at that precise moment it is necessary to change the environmental variables **PATH** and **LD_LIBRARY_PATH**.

It is important to notice that in order to debug the CUDA part of this program some changes have to be made to the CUDA code auto-sufficient to be able to debug and profile (e.g. needs to have a **main** function). This is due to the fact that the CUDA file that is used in this CUDA-IDL implementation is prepared to be called by the IDL part of the program. Note that a GPU that is running on Linux X Window System (X11), a windowing system for bitmap displays, cannot be used to debug a CUDA application, which means that the GPU must be exclusively used for debugging purposes. In case the system contains a single GPU, a possible solution is to remotely access the system through Secure Shell (SSH), a protocol for securely getting access to a remote computer. Another way to debug is by printing computed data through the bash command line. In this case, since it is not possible to print data directly from the GPU, it is always necessary to transfer it back to the host and then print it to the command line (as it is done with a C program). In this method it is also possible to use CUDA-MEMCHECK simply by passing the application's name as a parameter to CUDA-MEMCHECK on the command line, as shown next:

cuda_memcheck ./application_name.out

A GPU that is running X11 can still be used for profiling GPU applications. In fact, the Nsight Profiler is an excellent tool to use after the program is running and producing the expected results, delivering vital feedback for optimizing the application. The use of **dmesg** may be useful when some problems are detected with **cuda_memcheck**. **dmesg** prints the message buffer of the kernel, which is useful to identify memory related problems.

For now, the CUDA code deals with a group of 88 x 56 bins per CUDA invocation, meaning that multiple CUDA invocations are needed in order to compute every single bin of the detector (2816 x 3584 bins). The general idea of the CUDA code is as follows: the determination of the intersections are parallelized by distributing the rays between available threads; for the calculation of the distances between intersections, each thread is responsible to calculate a distance between intersection n and $n + 1$.

The integration of CUDA in IDL was done through the use of LINKIMAGE. Please refer to [10] in order to understand first the IDL-C interface. Here are some of the key aspects from the IDL-C learning experience:

- LINKIMAGE C modules must be compiled and linked before the IDL execution – the C module has to include the location of the "export.h" header file in the compilation step, note that typically the header file is found in the idl "external" directory;
- every variable in IDL is in reality a C structure called *IDL_VARIABLE*;
- the access of IDL variables must be done through a C pointer called *IDL_VPTR*, in which the extracted values depend upon how it was created in IDL;
- in the data transfer between C and IDL one must consider that the representation of a data type may be different (e.g. integers in IDL are short C integers).

The general idea of the IDL code to incorporate the CUDA integration is the following: IDL receives the voxels' coordinates and respective distances from the CUDA code and applies the following instructions (Figure 1): check the z-coordinate of each two voxels and verify if they change from 60 (the maximum z) to 0 (the minimum z); if so, the algorithm cuts between those elements, grouping the voxels that correspond to the same bin. Finally, every voxel that corresponds to a single-bin is processed with an iterative reconstruction algorithm, in the same way as it would be done in the pure-IDL version. Note that this CUDA-IDL implementation uses a scale factor in order to have faster reconstructions, being considered that each bin possesses a dimension of 4, 8 or 16 times the original 0.085mm. This scaling results in a reduction of the necessary memory to be allocated at the same time.

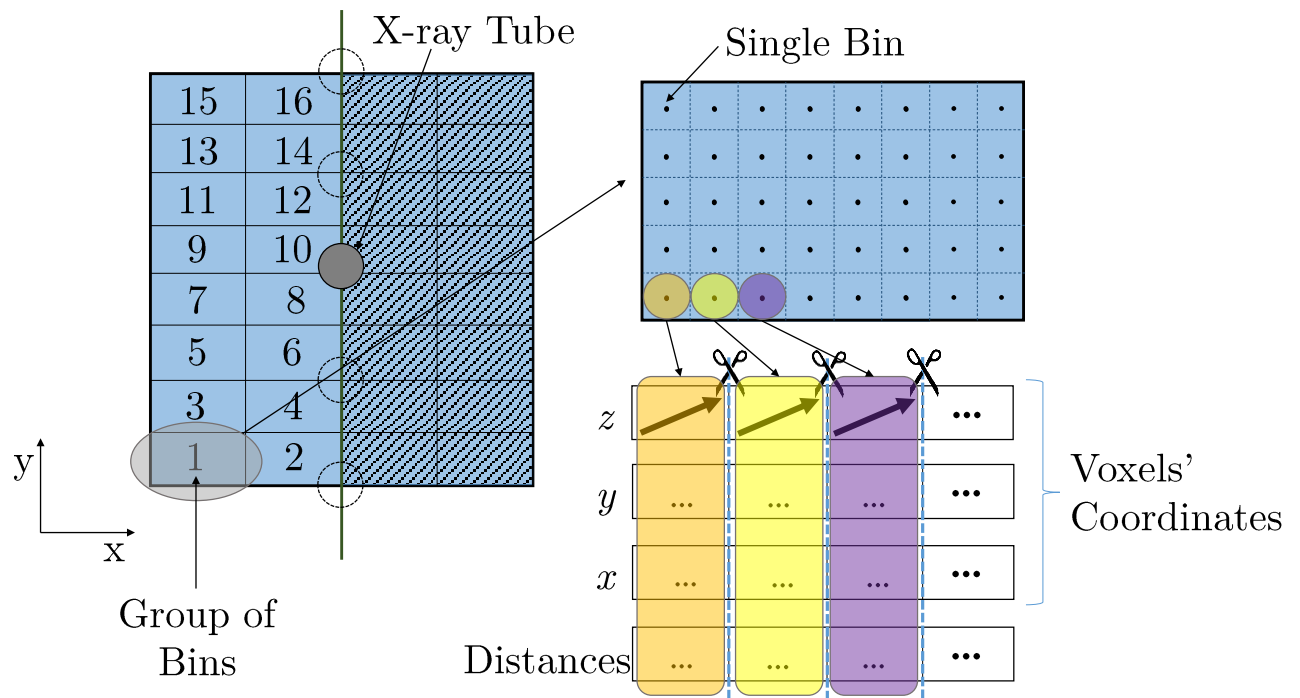


Figure 1 - Illustration of how the data received in IDL from the CUDA code (the voxels' coordinates and distances) are processed to organize it per bin.

The CUDA entry function is expecting to receive from each IDL call 6 variables:

- **x** – integer value that indicates the x position of the first bin of the group of 88 x 56 bins to be processed;
- **y** – integer value that indicates the y position of the first bin of the group of 88 x 56 bins to be processed;
- **angle** – float value that indicates the angle of the detector;
- **radius** – float value that indicates the radius of the detector;
- **bins** – vector of long values empty in the beginning, to be calculated in the GPU and returned to IDL with the voxels' coordinates (from this value it is possible to obtain each coordinate – x, y and z);
- **distArrTemp** – vector of float values empty in the beginning, to be calculated in the GPU and returned to IDL with the distances.

Concerning memory optimizations: firstly, it is necessary to consider that the Thrust library is being used and more particularly its function `sort_by_key`, which needs extra memory allocation; secondly, with a GPU with more memory several modifications may be done related with the number of bins processed for each time in the GPU (both the CUDA code and the IDL code can be easily changed to deal with more bins at the same time).

It may be useful to know how to disable the thread pool in IDL if one aims to study multithreading in the CPU [11] and compare it with the GPU results. This can be done with the following simple instruction inside IDL:

CPU, TPOOL_NTHREADS = 1

This instruction disables the thread pool for the session by setting the number of threads to use to one.

Support

For support please contact Pedro Ferreira, the creator of this program: pr.ferreira@campus.fct.unl.pt

Trademarks and Licenses

IDL is a registered trademark of Exelis Visual Information Solutions.

NVIDIA and CUDA are registered trademarks of NVIDIA Corporation in the United States and other countries.
NVIDIA

X Window System and X11 are trademarks of The X.Org Foundation.

References

- [1] J. Qi and R. M. Leahy. "Iterative reconstruction techniques in emission computed tomography". In: *Physics in Medicine and Biology* 51.15 (2006).
- [2] A. H. Andersen and A. C. Kak. "Simultaneous Algebraic Reconstruction Technique (SART): A Superior Implementation of the ART Algorithm". In: *Ultrasonic Imaging* 6.1 (1984), pp. 81–94.
- [3] M. N. Wernick and J. N. Aarsvold. *Emission Tomography: The Fundamentals of PET and SPECT*. Elsevier Science, 2004.
- [4] H. Hudson and R. Larkin. "Accelerated Image Reconstruction Using Ordered Subsets of Projection Data". In: *Medical Imaging, IEEE Transactions on* 13.4 (1994), pp. 601–609.
- [5] CUDA GPUs. NVIDIA. [Online]. Available: <https://developer.nvidia.com/cuda-gpus> (visited on Sept. 2, 2014).
- [6] CUDA Toolkit. NVIDIA. [Online]. Available: <https://developer.nvidia.com/cuda-toolkit> (visited on Sept. 8, 2014).
- [7] Thrust. NVIDIA. [Online]. Available: <https://developer.nvidia.com/Thrust>
- [8] Quadro FX 3800, NVIDIA. [Online]. Available: http://www.nvidia.com/object/product_quadro_fx_3800_us.html (visited on Aug. 20, 2014).
- [9] Quadro 600, NVIDIA. [Online]. Available: <http://www.nvidia.com/object/product-quadro-600-us.html> (visited on Aug. 20, 2014).
- [10] R. Kling, *Calling C from IDL: Using DLM's to Extend Your IDL Code*. Kling Research and Software, 2003.
- [11] Controlling the IDL Thread Pool. [Online]. Available: http://www.exelisvis.com/docs/Controlling_the_IDL_Thre.html#threading_2361397167_998627 (visited on Sept. 9, 2014).