EMBEDDED WAVELET IMAGE RECONSTRUCTION IN PARALLEL COMPUTATION HARDWARE

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

September 2015

By
Jorge Guevara Escobedo
School of Electrical and Electronic Engineering

Contents

List of Tables

List of Figures

List of Acronyms

ASA Application Specific Architectures

ASIC Application Specific Integrated Circuit

AVERR Average Error

CAD Computer Aided Design

CPLD Complex Programmable Logic Device

CPU Central Processing Unit

CT Computed Tomography

CWT Continuous Wavelet Transform

DFT Discrete Fourier Transform

DSA Domain Specific Architectures

DSP Digital Signal Processor

DWT Discrete Wavelet Transform

ECT Electrical Capacitance Tomography

EIT Electrical Impedance Tomography

ERT Electrical Resistance Tomography

FBP Filtered Backprojection

FFT Fast Fourier Transform

FIR Finite Impulse Response

FOV Field of View

FPGA Field Programmable Gate Array

FST Fourier Slice Theorem

FT Fourier Transform

FWT Fast Wavelet Transform

GPA General Purpose Architectures

GPU Graphics Processing Unit

GT Gabor Transform

GUI Graphical User Interface

HDL Hardware Description Language

HHT Hilbert-Huang Transform

IP Semiconductor Intellectual Property Core

ISE Integrated Synthesis Environment

LUT Look-up Table

MAC Multiply-Accumulate

MRI Magnetic Resonance Imaging

MSE Mean Square Error

NABS Normalised Absolute Error

PC Personal Computer

PET Positron Emission Tomography

PSNR Peak Signal to Noise Ratio

RAM Random-Access Memory

RH Reconfigurable Hardware

ROI Region of Interest

ROM Read-only Memory

RT Radon Transform

SPECT Single-photon Emission Computed Tomography

SSIM Structural Similarity Index

STFT Short-Time Fourier Transform

QMF Quadrature Mirror Filtering

VHDL VHSIC Hardware Description Language

WFT Windowed Fourier Transform

WT Wavelet Transform

Abstract

The University of Manchester, September 2015. Abstract of thesis submitted by Jorge Guevara Escobedo for the Degree of Doctor of Philosophy (PhD) entitled "Embedded wavelet image reconstruction in parallel computation hardware".

In this thesis an algorithm is demonstrated for the reconstruction of hard-field Tomography images through localized block areas, obtained in parallel and from a multiresolution framework. Block areas are subsequently tiled to put together the full size image. Given its properties to preserve its compact support after being ramp filtered, the wavelet transform has received to date much attention as a promising solution in radiation dose reduction in medical imaging, through the reconstruction of essentially localised regions. In this work, this characteristic is exploited with the aim of reducing the time and complexity of the standard reconstruction algorithm. Independently reconstructing block images with geometry allowing to cover completely the reconstructed frame as a single output image, allows the individual blocks to be reconstructed in parallel, and to experience its performance in a multiprocessor hardware reconfigurable system (i.e. FPGA). Projection data from simulated Radon Transform (RT) was obtained at 180 evenly spaced angles. In order to define every relevant block area within the sinogram, forward RT was performed over template phantoms representing block frames. Reconstruction was then performed in a domain beyond the block frame limits, to allow calibration overlaps when fitting of adjacent block images. The 256 by 256 Shepp-Logan phantom was used to test the methodology of both parallel multiresolution and parallel block reconstruction generalisations. It is shown that the reconstruction time of a single block image in a 3-scale multiresolution framework, compared to the standard methodology, performs around 48 times faster. By assuming a parallel implementation, it can implied that the reconstruction time of a single tile, should be very close related to the reconstruction time of the full size and resolution image.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I want to thank my parents, my sister and my grandmother who have always unconditionally supported me, and who along with my nephew, are my everyday motivation. To my family in general for their all time best wishes.

I want to thank my Supervisor, Professor Krikor B. Ozanyan, for encouraging me through the most difficult moments and for his support and guidance during the development of this research.

I want to thank José for being a great friend, and for his willingness to transmit his knowledge and experiences, as well as to my other friends within the SISP Group: Solon, Noushin, Xie, Miguel and Shahab.

Also many thanks to my friends in Manchester: David, Chava, Ivette, Michelle, Leo, Freddy, Botello, Fabiola and Daniel, with whom I have shared so many great moments and from which I have always received unconditional support during my stay in Manchester.

Thanks to CONACYT, who through its work and management of Mexican citizens' contributions, make it possible to provide all us, Mexican students, with the necessary economic support to continue with our postgraduate studies.

Chapter 1

Introduction

1.1 Tomography

Obtaining a visual representation of an object's inner-structure is a common problem in a wide range of scientific and industrial processes. The solution to such a problem was the main motivation that led Tomography to emerge as promising field of study. In the literature, Tomography imaging tends to refer to the image formulation of an object, from either emitted or transmitted electromagnetic radiation data collected from a certain number of different directions [1]. As every different direction is specified by an angle, a set of collected energies within the same direction, constitute an angular projection. The set of angular projections that map the object in the range between 0° to 180°, is the Radon Transform (RT) of that object [2].

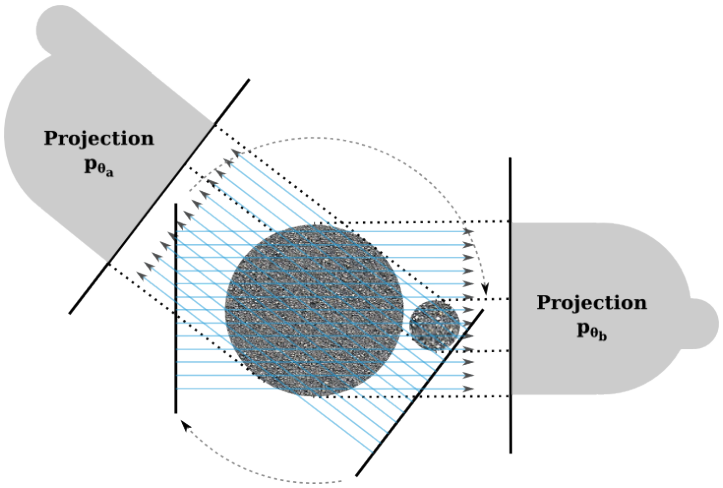
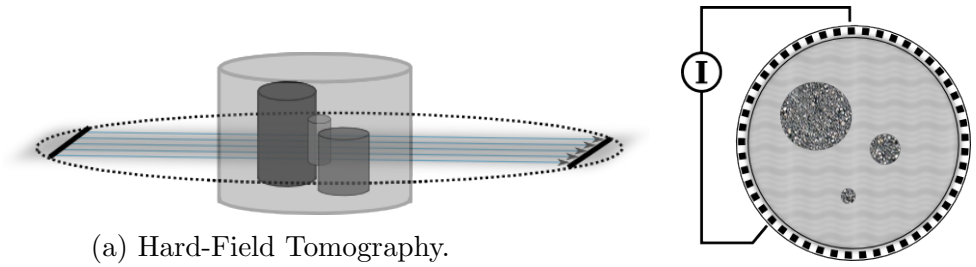


Figure 1.1: Angular Projections.

In 1917, Johan Radon found the solution to the problem of how to reconstruct an image from its angular projections [3]; Radon's work later became the mathematical foundation for Tomography. In 1964, Allan Cormack employed Radon's work in the development of an algorithm, that took into account the difference between tissue densities, for the assembling of images [4]. A few years later, Godfrey Hounsfield extended Radon's work to an experimental implementation; the first X-ray Computed Tomography (CT) scanner [5]. Hounsfield's invention supported by Allan Cormack's mathematical algorithms, showed together that it is possible to generate high-quality cross-sectional images from measured line integrals. Such an achievement set the realisation into practice of an engineered instrument, from the application of several scientific concepts. Hounsfield-Cormack's contribution was significant enough, not only to earn the Nobel prize in 1979, but also to set up a revolutionary impact in the practice of Medicine [6].

1.1.1 Hard-Field vs. Soft-Field Modalities

According to the propagation of energies across the object being imaged, Tomography can be classified into two main categories: hard-field and soft-field. In hard-field Tomography the source energy travels in a constant direction, so collected values depend only on the interactions along a defined path. Some examples of hard-field modalities are: X-ray CT, Positron Emission Tomography (PET) and Magnetic Resonance Imaging (MRI). Figure 1.2 helps to illustrate the difference between both modalities.



(a) Hard-Field Tomography.

(b) Soft-field Tomography.

Figure 1.2: Tomography categorisation in terms of energies propagation.

Soft-field modalities are more complex than the hard-field counterpart in the way in which source energy propagates across the whole probed object. In soft-field modalities, measured values are distributed throughout the entire

volume, and are dependent on the physical properties of such volume [7]. Electrical Resistance Tomography (ERT), Electrical Capacitance Tomography (ECT) and Electrical Impedance Tomography (EIT) are some examples of soft-field modalities.
The algorithm developed and presented throughout this theses, belongs to the hard-field modalities category.
The pursuit to create a non-invasive alternative to look inside patients' bodies, put Tomography on a constant trend toward faster reconstructions associated with image accuracy enhancement [6]. As a consequence, regardless Tomography was mainly enforced as a solution in medical imaging, and its development has been expanded into industrial application areas [8]. Nowadays a vast variety of modalities can be encountered in a wide range of different applications.

1.1.2 Tomography Image Reconstruction Algorithms

Classic reconstruction algorithms were mostly developed during the research boom in the 1970s, soon after Hounsfield-Cormack's invention. Nowadays, given the innumerable different algorithms and variations of themselves, that have appeared, it is impossible to make a concise and reliable classification. Nonetheless, in a very broad sense, algorithms for image reconstruction from angular projections, are characterised either as transform methods or as series expansion methods [9].

1.1.2.1 Transform Methods

Transform reconstruction methods rely on the direct application of formulas that are derived from the theory foundations and models. This kind of methods perform reconstruction in a single step. The Filtered Backprojection (FBP), which is derived from the RT, is a clear example of an analytical method [2].

1.1.2.2 Series Expansion Methods

In series expansion methods, the image to be reconstructed is assumed as a matrix of unknown values. This matrix is given with an initial estimate of the object's attenuation distribution, which is then used to construct a system of linear equations, in terms of the trajectories followed by projections during data

acquisition. The system of equations serves to find the contribution of every matrix value over the reconstruction grid [9].
Unlike the transform methods, series expansion methods perform the reconstruction in several steps until reaching consistency between calculated and measured projections [1, 10, p.57].

1.1.3 Incomplete Data Problem in Tomography

The most important reconstruction method in Tomography is the FBP. The FBP is among the preferred methods in real implementations and has been proved to deliver accurate images at high computational efficiency [11, 12]. Such method is a generalisation of the inverse RT formula for a practical implementation and consists of two main elements: a filtering stage (usually ramp filtering) to amend the negative effects caused by the gathering of values in the Fourier space dc term (by means of the Fourier Slice Theorem), and the backprojection stage, which is the reordering of filtered projections back over the reconstruction grid [1, p. 65].
However, as being an analytical method, the FBP suffers from the drawback, that in its formulation, noise is not taken into account and measured projection data is assumed to be continuous and acquired from a certain standard geometry. In other words, for its proper operation, the FBP is conditioned to the availability of quality projection data: acquired at sufficient number of angles and equally spaced between the range from 0° to 180°.
Noise related problems are commonly alleviated through the utilisation of post-filtering operations, as well as the use of combined projection data obtained from multiple scans [13]. For the incomplete data problem two common sources that drive projection data to incompleteness are identified, both of them originated at the acquisition set-up: limited angle projections and truncated projections [14].
Because of effort in understanding the nature of projection data incompleteness date back from as early as the first CT algorithms development era [15], substantial research has been produced based on two different types of practical approaches [16–19]. The first approach refers to the estimation and compensation of the missing data (e.g. [20]). The second approach aims to reduce the bad effects that result when only available data is considered in the reconstruction; this approach depends on the generalisation of existing

algorithms from the underlying theory (e.g. [21–23]).

1.1.3.1 The Interior Problem

The choice of the approach to be employed depends on two different situations: when data incompleteness is caused by technical restrictions, so it is undesired, and when data incompleteness is intentionally induced in the pursuit to limit the reconstruction to a specific area, also known as Region of Interest (ROI) [24]. The problem of reconstructing a ROI arose in medical imaging, due to the constant trend on reducing the overall radiation exposure to patients [25, 26]. The reconstruction of a ROI falls in a case where acquisition from full angle view is achievable, but projections are composed only from rays passing over an specific area, so projection data is truncated; this case receives the name of interior problem [19, 22]. This is illustrated below in Figure 1.3.

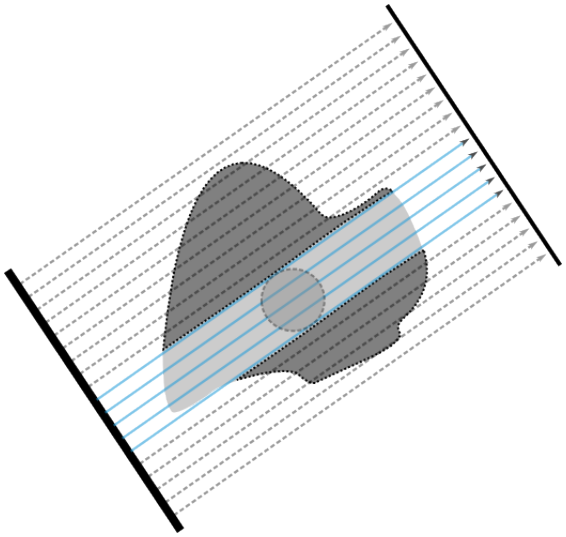


Figure 1.3: Data acquisition over a Region of Interest

Naturally, the FBP in its standard form, is not considered as a valuable method to be considered in an interior problem reconstruction. The FBP is derived from the RT, so its global dependency relies in a property of the RT that states: the inversion formula for the RT is globally dependent upon the projection data set, so any missing amount of data, will exhibit undesirable effects in the overall image [23].
The global dependency of the FBP is attributed to the discontinuity in the Fourier domain that the ramp filter introduces in the filtering stage of the FBP.

Such discontinuity is the cause of the support spread in space domain, of projection data after being filtered [27]. Therefore, most of the methods dedicated to treat the interior problem aim at "reconstructing discontinuities" [17].
Some examples of approaches to tomographic image reconstruction that belong to the interior problem case, are: Λ-Tomography [21], Pseudo local Tomography [28], Two-Step Hilbert Method [29], and Wavelet-based Local Tomography [23, 30–34].

1.2 Wavelet-Based Tomography Image Reconstruction

1.2.1 The Wavelet Transform

The Wavelet Transform (WT) is a mathematical tool that arose as an extension of the Fourier Transform (FT) for the analysis of non-stationary signals. The main characteristic that makes the WT different from the FT, is that WT basis functions are not only localised in frequency/scale, but also in time/spacial-location. By definition, a wavelet function is a small oscillation confined into a finite interval, with zero value integral. Wavelet functions are dilated and translated to adapt to the different components of a signal; small support wavelets are good to look at high frequency components, while large support wavelet are employed to analyse low frequency components. Therefore, the WT makes it possible to study a signal at different resolutions: a coarse resolution provides a gross approximation of the signal, whereas higher and higher resolutions allow to see increasingly fine details of the signal. This process is called multiresolution.
The optimal choice of wavelet functions is done based on different properties, which among the most important are: the kind of basis that the chosen wavelet functions constitute, the regularity which indicates how smooth the wavelet is, and the number of vanishing moments, which are linked to the number of oscillations, and hence proportional to the size of the wavelet support [35, 36].

1.2.2 The Fast Wavelet Transform and The Multiresolution Representation

A different WT scheme, developed by Mallat in the early 1980s [37], is the Fast Wavelet Transform (FWT). The FWT is a concise and fast method whose resulting transformed data, has the same number of points as the original signal, is a linear process and allows perfect reconstruction. In practice, the procedure consists on convolving the signal with both a high pass and low pass filter to split the signal into a scaling and a wavelet coefficient, also referred to as approximations and details. Approximations wavelet coefficient, corresponds to the low pass filter output, which is a smoothed version of the original signal seen at half resolution with as half as many samples. Details wavelet coefficient, contains the higher frequencies that would have to be added to the approximations to recover the original signal, again with half as many samples. If the transform procedure is iteratively reproduced, approximations wavelet coefficient becomes the input to a new decomposition, producing new details and new approximations, being the latter an even more smoothed signal with a quarter as many samples than the original signal. By iteratively computing the FWT, it is possible to obtain a multiresolution representation of the signal, constituted by wavelet coefficient components that differ in size by a factor of two, as shown below in Figure 1.4 [38, p. 32].

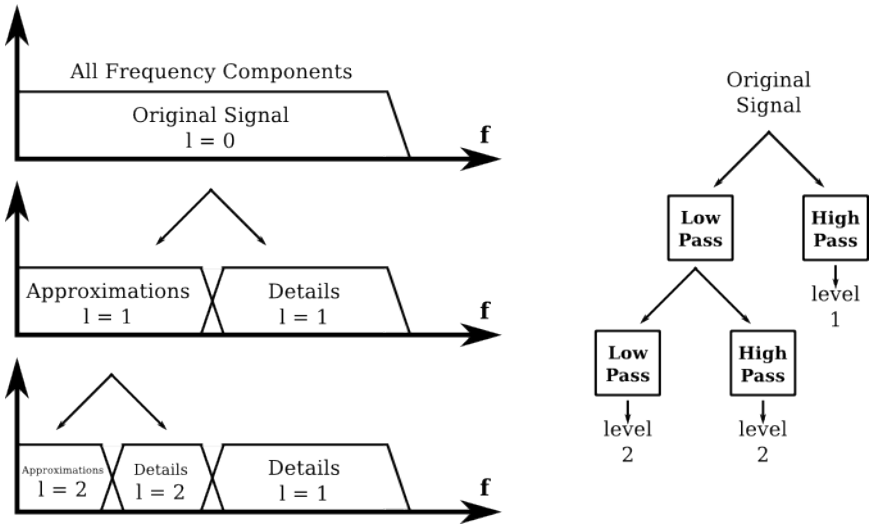


Figure 1.4: Multiresolution Decomposition

1.2.3 Wavelet-Based Image Reconstruction of a Region-of-Interest

According to [27], to overpass the discontinuity of the ramp filter in the FBP method and being able to accurately reconstruct a ROI, a set of basis functions, essentially compactly supported and constructed with enough zero moments, are required. If properly chosen, wavelet functions can fulfil the required properties, so if employed in the filtering stage of the FBP, projection data can remain compactly supported.
The FWT, and its extension to two dimensions which is explained in detail in Chapter 3, has been successfully employed in the FBP method. By generalising the FBP with the FWT, the method turns from the reconstruction of an image to the reconstruction of wavelet coefficients at different resolutions. Through this generalisation of the FBP, the backprojection stage is simplified because the computation (i.e. smearing) of filtered projections is done over sub-sampled grids [33, 34, 39].

1.3 Fast Tomographic Reconstruction Methods

A different trend in the development of image reconstruction methods, apart from accuracy and/or resolution quality, is the overall speed-up of the reconstruction process. As previously mentioned, the FBP is the method of choice when noiseless and complete projection data is available. However, the arising of relative technologies to the acquisition of data at faster rates, has encouraged the development of faster reconstructions aiming at filling the gap between data acquisition and image reconstruction [40, 41]. In [40] S. Basu and Y. Bresler identify two categories of existing fast algorithms: Fourier-based and Hierarchical.
Fourier-based algorithms come from the Fourier Slice Theorem (FST). These kind of algorithms consist on the transformation from polar to Cartesian of projection data in the Fourier space. Therefore, in Fourier-based methods, the image is reconstructed directly through the inverse two-dimensional Fast Fourier Transform (FFT) [42].
Hierarchical algorithms, as opposite to the Fourier- based counterpart, act directly either over the projection data domain, or image domain. The idea

behind hierarchical approaches is to split the problem of reconstructing a full size image into smaller problems; smaller-sized images reconstructed from fewer projections (i.e. divide and conquer) [43–46].
In [41] T. Rodet mentions a third category concerned to the speed-up of the backprojection step, which is the most computationally expensive operation within the FBP reconstruction process. An example of this approach is that all projections are computed at the same time in a parallel fashion [47–49].

1.4 Hardware Architectures and Tomography Implementations

Without distinction, for the achievement of requirements in terms of speed and/or resolution, any of the previously mentioned fast reconstruction algorithm approaches, is dependent on the computing platform employed in its practical implementation. From a truly computational point of view, in terms of flexibility and performance, computer system architectures are categorised in three main groups: General Purpose Architecturess (GPAs), Domain Specific Architecturess (DSAs) and Application Specific Architecturess (ASAs) [50, pp. 1-12].
ASAs, commonly in the form of an Application Specific Integrated Circuit (ASIC) can be encountered since the early days of Tomography development. ASICs involve dedicated designs through functional units that can be implemented in parallel, so they provide better performance for a specific class of application. Although, its high performance is obtained at the expense of very low flexibility, in as much as ASICs based design can not be modified after being manufactured. Another inherent drawback is related to the high costs that involve ASICs production, so they are only affordable by manufacturers of equipment that target large markets (e.g. [51–53]).
GPAs lie at the opposite side, Central Processing Unit (CPU) based implementations are the most flexible options, but the ones with the less performance. In Tomography, GPAs are mostly employed by researchers and/or common users to experience in new algorithms and acquisition geometries [41]. It is also common to encounter CPUs within the design flow of more specialised implementations. The low performance of general-purpose architectures is attributed to the fact that applications are adapted to the hardware, so every

piece of software run according to an instruction cycle that through a control unit manages accesses to memory, as well as operations of an arithmetic-logic unit [50, p. 3]. Nonetheless, CPUs have evolved to processors having several core units in a single chip device, capable to execute multiple tasks in parallel [54, 55]. Multi-core clusters are flexible low cost alternatives to high performance ASIC devices, although they are limited by communication speed (e.g. [56]).

DSAs have been also widely employed in Tomography with the purpose to speed-up custom reconstruction algorithms. DSA devices are based on the general-purpose model, but with elements adapted for a certain class of algorithms, so they can deliver better performance [50, p. 5]. Among the preferred DSA devices is the Graphics Processing Unit (GPU), which is a device created under the stream processing paradigm [54, 57]. Stream processors are the approach created to fill the gap between flexible GPA solutions and high performance ASA approaches. Stream processors are programmable devices, they allow high number of computations to occur in parallel, and are optimised to use minimal global communication and storage [58]. Some examples of Tomography implementations in DSAs are in [54, 59–63].

Nonetheless DSA architectures preserve the flexibility of the GPA model and can reach better performance, they still being the class of architectures in which the application is adapted to the hardware. Reconfigurable Hardware (RH) is a fourth alternative for the acceleration of computationally intensive algorithms. This kind of architecture is closer to the ASA model in the sense that hardware is adapted to the application, so the application design can be based on functional units for specific application tasks running in parallel. Additionally, RH allows to modify all or part of its structure at compile or execution time [50, p. 9]. Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) are the most representative [55]. Nonetheless FPGAs cannot deliver the same performance as an ASICs, they have been increasingly preferred over ASAs given its flexibility, low cost and profit for rapid prototyping [64, 65]. In Tomography, FPGAs have been employed in the design of custom reconstruction algorithms, through dedicated hardware elements running in parallel (e.g. [55, 66–73])

The algorithm presented in this thesis has been designed to be implemented as a set of custom functional units performing different tasks in parallel, so the

reconfigurable hardware approach, has been chosen as the platform to experience with the designed algorithm parallelism.

1.5 Application Field

From the brief background theory presented in this chapter, as well as the more detailed concepts covered in Chapter 2, it can be noticed that the fundamentals of image Tomography reconstruction employed in the development of the proposed algorithm, belong to X-Ray CT. Nonetheless, it is important to state that this work has been developed with theoretical purposes and in its current state, does not have a specific target application. The latter is mentioned with the aim to specify that the proposed reconstruction algorithm does not depend upon an acquisition modality, and hence is not particular of any application field (i.e. medical imaging, industrial process Tomography).
A main topic of study in the development of the work presented in this thesis, is the work related to the utilisation of the WT to achieve the accurate image reconstruction of localised areas belonging to a ROI. Such a developed work was mainly motivated by the need to reduce the radiation dose exposure to patients in medical imaging, where X-Ray CT was a mainstay [27, 32–34, 74]. Nonetheless, the same principles have been applied in other application fields; on one side it is possible to encounter complex X-Ray CT systems in industrial environments (i.e. for the measurement of gas and liquid hold-up in chemical vessels and micro structures) [75–79]. On the other side it is possible to encounter wavelet-based ROI image reconstruction applications using THz as the Tomography modality [80–82].

1.6 Research Aims and Objectives

This proposed approach is a generalisation of the FBP transform method that aims to take advantage of the wavelet transform properties, used to perform reconstruction of ROIs from truncated projection data [27].
The main idea is to intentionally adapt acquired data in the projection domain, to independently and in parallel, perform reconstruction of several ROIs. The proposed approach resembles the objective of the hierarchical fast reconstruction concept, in the sense that wavelet treatment of projection data is used to split

the inverse problem into a set of smaller and less complex problems.
In order to achieve such a proposed approach, the following objectives were defined:

- Accomplish a proper implementation of the standard FBP image reconstruction method, based on the MATLAB *iradon* function.
- Modify the filtering stage of the standard FBP by substituting the ramp filter by a WT-based filter, followed by the achievement perfect reconstruction.
- Perform local reconstruction from truncated data and analyse the results.
- Generalise the WT-based FBP to a multiresolution decomposition framework.

In addition, as consequence of the multiresolution representation provided by the WT, the reconstruction process turns from inverting a full-size full-resolution image, to a set of less-dense partial-resolution wavelet coefficient images [33].
This reformulation of the reconstruction method accomplishes one of the proposed algorithm aims, which is to improve the computation efficiency of the backprojection step in the FBP. The backprojection step is considered to be the most computationally expensive element within the FBP method, therefore its efficiency optimisation belongs to the aim for fast reconstruction.
In order to improve the backprojection operator the following objectives were defined:

- Evaluate the speed gain derived from the multiresolution decomposition of projection data by using the modified ramp filter.
- Achieve perfect reconstruction of projection data decomposed into several scales.
- Achieve a parallel implementation of the projection data WT decomposition and synthesis.

This way, through the combination of both approaches, the parallel reconstruction of several ROIs in a multiresolution framework was completed.

Finally, in order to experience with a true parallel implementation, the designed algorithm was refactored into a set of Hardware Description Language (HDL) components having the aim to achieve the computation of a projection angle, which represents the minimum independent processing unit within the parallel system.
In order to obtain a hardware architecture belonging to the processing of a single projection angle, intended to be replicated according to a set of input parameters, three different components were idealised:

- The filtering of projection data.
- The backprojection operator.
- The parallel inverse two-dimensional FWT.

Each of these hardware architecture equivalents were constructed and validated at input/output level based on the MATLAB algorithm.

1.7 Contributions

The main contribution of the work presented in this thesis is the combined algorithmic/hardware formulation, of a fast parallel reconstruction method, designed to be employed with hard-field modalities, and projection data acquired through a parallel-beam geometry acquisition set-up.
Developing such a fast parallel reconstruction algorithm involved the achievement of the objectives defined and mentioned in the previous section, from which the following contributions were identified:

- The complexity reduction in the computer implementation of the standard FBP derived from the analysis of the Fourier domain data symmetry, explained in Section 4.2.
- The accurate image reconstruction and speed gain achieved in Section 4.3 from the modified wavelet-based FBP, by exploiting the overlapping between projection data and wavelet filters within the Fourier space.
- The formulation of a parallel two-dimensional FWT for the analysis of projection data, as well as synthesis of the reconstructed coefficient images, at different scales. This approach considerably improved the

reconstruction speed as well as increased the parallel granularity of the algorithm, as shown in Section 4.4.

- The development of a simplified methodology shown in Subsection 4.5.1, to extract ROIs data in Radon domain from the complete projection data set.

- The development of a Tomography image reconstruction algorithm combining both main approaches achieved by employing the FWT: the accurate reconstruction of ROIs from truncated data, and the parallel multiresolution framework covered and explained in Section 4.5.

- The implementation in a HDL platform of the minimum replicable element within the parallel system; the computation of a single projection data angle. This is covered in Chapter 5.

1.8 Thesis Outline

This thesis presents the design of a fast parallel image reconstruction algorithm, which formulation relies in the achievements related to the accurate image reconstruction from truncated data, by using the WT. In this chapter, Tomography fundamentals are briefly covered, followed by an overview of the Tomography inverse problem and its most common solution approaches. Emphasis is put in the FBP reconstruction method, from which the pursuit to find a solution to weaken its dependence of vast and high quality projection data, led to the development of fast reconstruction generalisations.
Chapter 2, which is dedicated to the Tomography image reconstruction problem, goes beyond the brief concepts addressed in the Introduction, and covers the RT mathematical foundation, that evolved up to the creation of practical inversion methods.
Chapter 3 presents a review of the origins and efforts that led to the creation of the WT. The chapter not only explains the WT utilisation, but also the differences between each of its versions, along with the benefits and disadvantages found in each of them. The properties that made the WT an attractive tool, to be employed in Tomography image reconstruction, are as well exhibited.

Chapter 4 describes the procedure involved in the design of the image reconstruction parallel algorithm, which covers the computer implementation of the standard FBP, its customisation for a better performance, the inclusion of the WT in its formulation, and two different parallel proposed approaches to reach fast image reconstruction. Results in terms of quality and reconstruction time, are presented for each one of the explained parallel approaches.
In Chapter 5, the implementation of the wavelet-based, parallel reconstruction algorithm, in a parallel computation hardware (i.e. FPGAs), is documented. A discussion about the experimental results, as well as its feasibility, is included.
Chapter 6 gathers the results and observations, collected along the development of this research, into final conclusions. Future work is as well discussed within this chapter.

Chapter 2

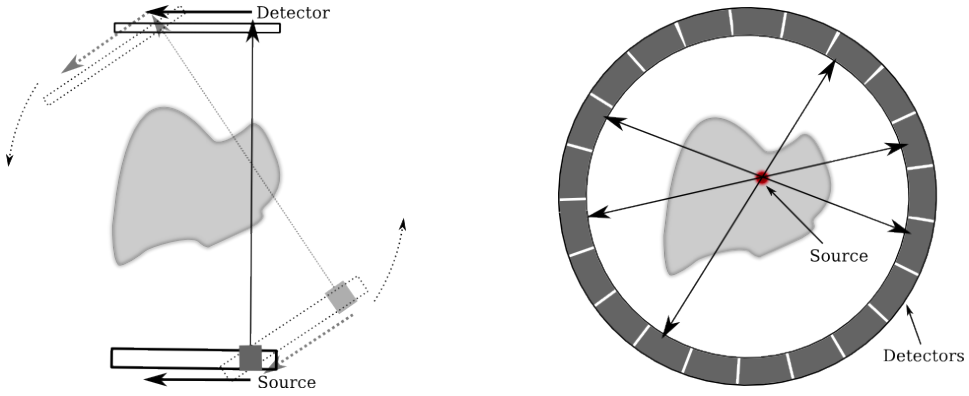
Tomography Image Reconstruction

The word Tomography is composed by the junction of the Greek words *tomos* (i.e. slice or section) and *graphein* (i.e. drawing); in English it is interpreted as "draw a section". Tomography is one of the existing techniques for the cross-sectional imaging of an object from either emitted or transmitted electromagnetic radiation [1]. Nowadays, not only the literature related to Tomography is largely vast, but it is possible to come across Tomography in a wide range of different applications. Most of the achieved progress in Tomography has been boosted by demands in medicine and industry, on a constant trend in obtaining faultless images at high speed rates [6].

2.1 Transmission Data Tomography

The main goal of hard-field Tomography modalities is the reconstruction of cross-sectional images, representing the inner density distribution of an object, from projection data acquired at different angles within the range from 0° to 180°.
In hard-field modalities, the imaging system acquisition is determined by the location of the radiation source. When the radiation source is placed outside the object, transmission data is acquired as shown in Figure 2.1a. Conversely, when the radiation source resides within the object, acquisition is done from an emission source (Figure 2.1b).
This chapter is dedicated to explain the methodology involved in the

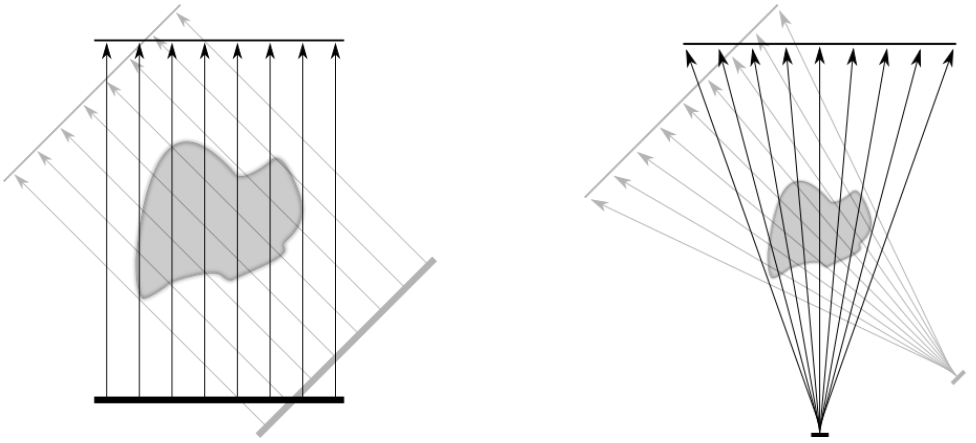
Tomography related acquisition and image reconstruction, for the transmission data case. The principles here exposed are based on (but not limited to) which is probably the most common example found in literature to understand the acquisition set-up of transmission hard-field modalities, the X-ray CT.



(a) Transmission Data Acquisition. (b) Emission Data Acquisition.

Figure 2.1: Imaging System Acquisition.

In CT, the X-ray beam is considered as a set of lines (i.e. rays) arranged in a regular pattern, commonly referred as the acquisition geometry [11]. In a primary classification, the acquisition geometry is divided in two categories: parallel and divergent, as shown in Figure 2.2.



(a) Parallel Beam. (b) Divergent Beam (Fan Beam).

Figure 2.2: Acquisition Geometries.

In a subsequent classification, the divergent geometry is referred as fan beam, for a two-dimensional data acquisition (Figure 2.2b), and cone beam (Figure 2.3) for three-dimensional data acquisition case.

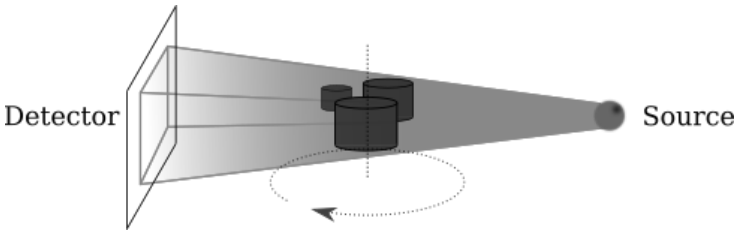


Figure 2.3: Cone Beam Acquisition Geometry.

A beam ray with an initial intensity I_0 is attenuated while travelling across an object, and the output intensity I measured by a detector. Measurements taken from intensity losses of several rays within a beam, acquired at different angles, are gathered to form a projection data set. The intensity loss of radiated energy passing through an object, is mathematically modelled by the Beer-Lambert Law [11].

$$I = I_0 \cdot e^{-\int_L \mu(x)dx}. \tag{2.1}$$

In (2.1), I refers to the ray intensity recorded at the detector, once the ray has passed through the object, I_0 is the ray intensity at the initial point (i.e. source), $\mu(x)$ is the linear attenuation coefficient of the object at a certain point x, and L is the path followed by the ray from the source to the detector.
The symbol μ refers to the attenuation coefficient, which is a property of the material and varies with respect to the incoming radiated energy in the following manner: μ becomes smaller when the intensity of radiated energy gets higher, and on the contrary, μ becomes higher when radiated energy suffers attenuation. It is important to mention that in (2.1) the attenuation due to interference, scatter, and/or absorption of energy, is not considered. Moreover, Equation 2.1 is only valid for the simplest case, in which the ray is assumed as a monochromatic continuous straight line as shown below in Figure 2.4.

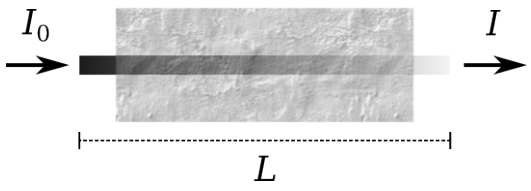


Figure 2.4: Continuous Attenuation Model.

For a practical implementation, instead, the attenuation coefficient μ is not constant along the distribution of the object, so it is discretised in a set of

isometric regions (i.e. pixels), with different attenuation factors. This is illustrated below in Figure 2.5.

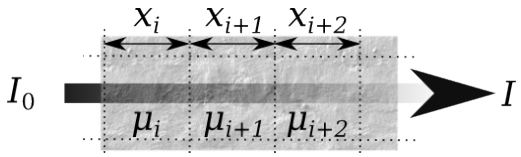


Figure 2.5: Discrete Attenuation Model.

The following expression is the discrete version of 2.1, where μ_i is represented as the attenuation coefficient along the distance x_i.

$$I = I_0 \cdot e^{-\sum_{i=1}^{i=N} \mu_i \cdot x_i}. \tag{2.2}$$

Again, from the continuous version in Equation 2.1, it is possible to see that the intensity loss is modelled by a line integral along the path L. If the problem is extended to a higher dimension, by considering the energy travelling through L, but across a two-dimensional plane, with attenuation coefficient $\mu = \mu(x, y)$, the intensity loss is then obtained through:

$$I = I_0 \cdot e^{-\int_L \mu(x,y)ds}. \tag{2.3}$$

If Equation 2.3 is rearranged to form an expression as the logarithm of the input/output intensity ratio, a line integral of the attenuation coefficient of a two-dimensional plane, along the path L is obtained.

$$r = -ln\left(\frac{I_0}{I}\right) = \int_L \mu(x, y)ds. \tag{2.4}$$

By looking at Figure 2.1a, it is possible to see that displacing the source and detector in a perpendicular direction to L, a set of parallel line integrals is generated. Such set of lines constitute a projection at a certain angle. Either by rotating the source and detector or the object itself, projections at different angles can be obtained. So by using the expression below, it is possible to obtain line integrals in terms of ρ displacements, at an specific angle θ by:

$$r(\rho, \theta) = \int_L \mu(x, y)ds. \tag{2.5}$$

The process is illustrated in Figure 2.6, where θ is the rotation angle with respect to the reference x-axis, ρ is the perpendicular displacement of the ray, and $r(\rho, \theta)$ is the resulting angular projection. The acquisition of projection data from line integrals in terms of ρ displacements, and ordered by its angular position θ, is mathematically modelled by the RT, covered in the following section.

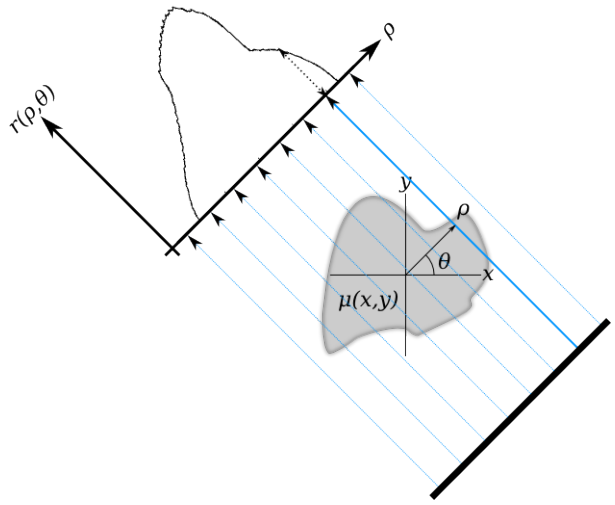


Figure 2.6: Angular Projection.

2.2 The Radon Transform

The RT was developed by the Austrian mathematician Johann Radon in 1917, who purely driven by theoretical interests, created a mathematical formula for the representation of a function in terms of its line integrals, so it could be possible to employ line integrals in the recovering of such function [3]. It was after fifty years of its creation, when in radio-astronomy, the RT was first employed as an imaging technique [83, 84]. The RT is the mathematical foundation of CT [85].

This section is dedicated to explain the RT, which in contrast to the mathematics involved in the acquisition of individual ray data, through the formulation of the Equation 2.5, the RT is a generalisation for the acquisition of a complete set of line integrals. A complete set, in a Tomography application, refers to the acquisition of the overall attenuation of radiation energy travelling

across an object. Therefore, in this case the object is modelled as a two-dimensional distribution of the energy attenuation coefficient $\mu(x, y)$, and a line integral refers to the total intensity loss of a beam as it travels the object across the path L [1].
The starting point is to make acquisition problem treatable by parametrising the set of line integrals in terms of the Polar coordinate system. This is done because such coordinate system is the most suitable to describe a problem with a circular symmetry. Parametrisation is performed by equations that, based on normal vectors to line integrals, satisfy every point along every ray path.

$$\rho = x \cos\theta + y \sin\theta. \tag{2.6}$$

Figure 2.7 shows how, by considering a parallel geometry arrangement, θ represents the angle between the x-axis and the rays' normal n. The displacement ρ, which is the normal distance from the origin to each ray, helps to distinguish each line from another; its sign differentiates rays having the same distance but in an opposite direction.

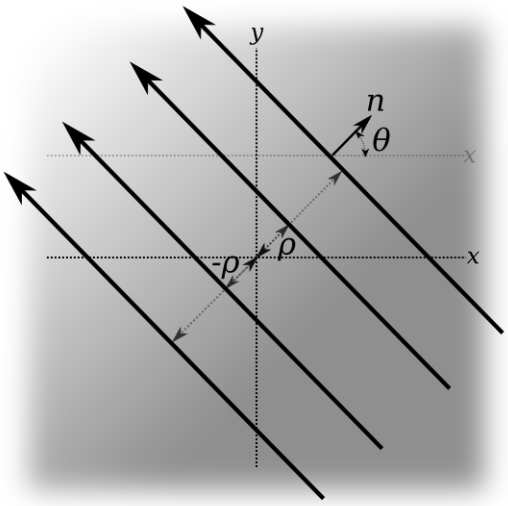


Figure 2.7: Parametrisation in Polar Coordinates

An important characteristic of the RT is that, given the mirror symmetry nature of the Radon space, only projections within the range from 0° to nearly 180° are necessary. Projection data beyond such range results redundant [86]. So, by leaving the parameter θ fixed and varying the displacements ρ, it is

possible to obtain a numerical representation of line integrals in the desired plane. This is formally expressed as the RT of $\mu(x, y)$, being $\mathcal{R}$ the RT operator.

$$r(\rho, \theta) = \mathcal{R}[\mu(\rho, \theta)] = \int_{L_{(\rho,\theta)}} \mu(x, y) dx dy. \tag{2.7}$$

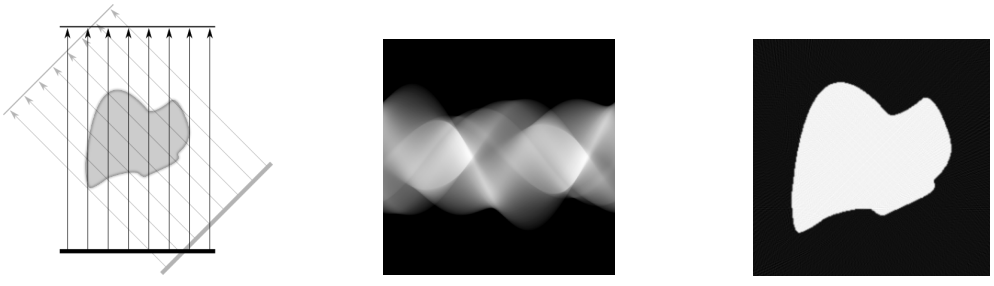
Moreover, if a distribution function δ is used to complement the parametric Equation 2.6, and obtain an expression indicating the impulse function concentrated along the path L.

$$\text{where: } \delta(\rho) = \begin{cases} \delta(\rho - x\cos\theta - y\sin\theta), \\ 1 \qquad x\cos\theta + y\sin\theta = \rho \\ 0 \qquad x\cos\theta + y\sin\theta \neq \rho \end{cases} \tag{2.8}$$

Equation 2.7, along with the δ distribution function 2.8, can then be expressed in the following form:

$$r(\rho, \theta) = \mathcal{R}[\mu(\rho, \theta)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mu(x, y)\delta(\rho - x\cos\theta - y\sin\theta) dx dy. \tag{2.9}$$

Equation 2.9 represents forward RT, the function that maps a two-dimensional distribution $\mu(x, y)$ to a set of line integrals $r(\rho, \theta)$, consecutively recorded whilst keeping track of angular position θ. The collection of the overall attenuation coefficients at different angular positions forms a sinogram (i.e. projection data set). The main objective of CT is to recover the two-dimensional distribution $\mu(x, y)$ from a sinogram $r(\rho, \theta)$. Figure 2.8 helps to illustrate the reconstruction stages.



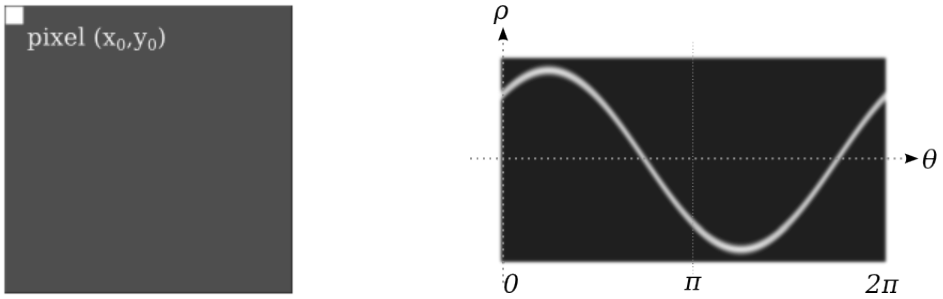
(a) Projection Data Acquisition.

(b) Resulting Sinogram.

(c) Reconstructed Image.

Figure 2.8: The Computed Tomography Process.

The term sinogram is attributed to the fact that the RT of a single pixel (e.g. in the (x_0, y_0) position within the pixel grid), generates a sinusoid curve as defined in the parametric Equation 2.6. It is then said that the process equals a point to curve transformation as shown below in Figure 2.9.



(a) Pixel Phantom. (b) Resulting Sinogram Curve.

Figure 2.9: Point to Curve Transformation.

The set of acquired projection data is therefore interchangeably referred as sinogram and represented as a two-dimensional function in the (ρ, θ) plane, as shown in Figure 2.9b.

2.3 The Inverse Radon Transform

From the previous section, it is possible to realise that in CT, the RT is the equivalent to a forward problem in the sense that it is the tool by which the acquisition of projection data is mathematically modelled. Although, the main objective of CT is to recover a visual representation from the object's projection data. Calculating the inverse RT is a more complex process, it is a mathematical problem that falls within the domain of inverse problems, and can be approached analytically through the FST [87].
The FST (i.e. Central Slice Theorem, Projection Slice Theorem) is the basis for transformed-based inverse methods. In two dimensions, the FST states that the one-dimensional FT of an angular projection $r(\rho, \theta)$ of a two-dimensional distribution $\mu(x, y)$ is equal to a slice through the origin of the two-dimensional distribution $M(\omega_x, \omega_y)$ within the Fourier space [88, p. 22]. This is illustrated in the figure below.

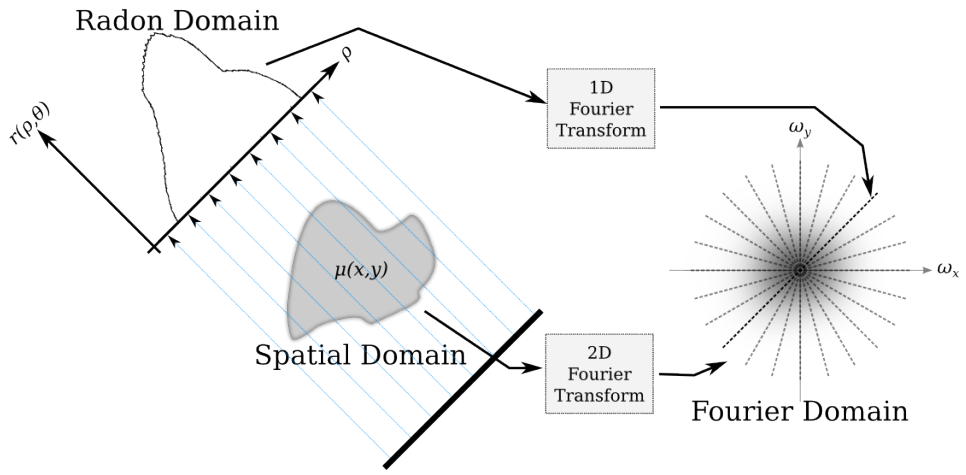


Figure 2.10: The Fourier Slice Theorem

The main idea is to fill the Fourier space $M(\omega_x, \omega_y)$ with projection data $r(\rho, \theta)$ acquired from as many as possible angular increments within the 0° to 180° range. Therefore, once $M(\omega_x, \omega_y)$ is ready, the two-dimensional distribution $\mu(x, y)$ of the projected object, is simply recovered by applying the two-dimensional inverse FT to $M(\omega_x, \omega_y)$.

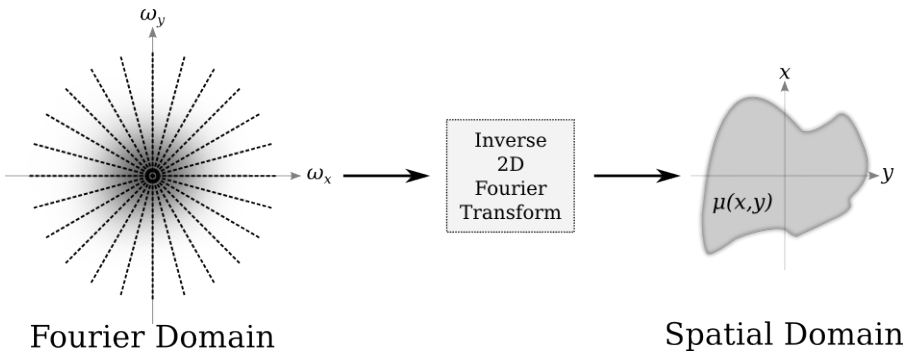


Figure 2.11: Image Recovery by Means of the FST

Mathematically, the FST is proven by first calculating the one-dimensional FT

of projection data $r(\rho, \theta)$ with respect to ρ and for a fixed angle θ.

$$\begin{aligned}\mathcal{F}_{1D}[r(\rho,\theta)] &= \int_{-\infty}^{\infty} r(\rho,\theta)e^{-j2\pi s\rho}d\rho, \\ &= \int_{-\infty}^{\infty} e^{-j2\pi s\rho}\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mu(x,y)\delta(\rho - x\cos\theta - y\sin\theta)dxdy\right]d\rho.\end{aligned} \tag{2.10}$$

After rearrange the integration order of terms within Equation 2.10, it follows:

$$\mathcal{F}_{1D}[r(\rho,\theta)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mu(x,y)\left[\int_{-\infty}^{\infty} e^{-j2\pi s\rho}\delta(\rho - x\cos\theta - y\sin\theta)d\rho\right]dxdy. \tag{2.11}$$

Equation 2.12 is the sampling property (i.e. sifting property) of the delta function, which can be applied to the integral with respect to ρ [89, pp. 74-77].

$$\int_{-\infty}^{\infty} f(x)\delta(x - x_0)dx = f(x_0). \tag{2.12}$$

Therefore, Equation 2.11 turns to the following expression:

$$\begin{aligned}\mathcal{F}_{1D}[r(\rho,\theta)] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mu(x,y)\left[e^{-j2\pi s(x\cos\theta + y\sin\theta)}\right]dxdy, \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mu(x,y)\left[e^{-j2\pi(xs\cos\theta + ys\sin\theta)}\right]dxdy.\end{aligned} \tag{2.13}$$

By definition, the two-dimensional FT $F(\omega_x, \omega_y)$ of a function $f(x, y)$ is denoted by:

$$\mathcal{F}_{2D}[f(x,y)] = F(\omega_x,\omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(x\omega_x + y\omega_y)}dxdy. \tag{2.14}$$

which is very similar to Equation 2.13, except for the sum $(xs\cos\theta + ys\sin\theta)$ of the exponential. Although, it can be seen that such term is equivalent to the dot product between the two-dimensional vector $\vec{x} = (x, y)$ and $(\cos\theta, \sin\theta)$. So, by rearranging the frequency variable s into the term $(\cos\theta, \sin\theta)$, the

two-dimensional frequency vector $\vec{\xi} = (s\cos\theta, s\sin\theta)$ is obtained. Equation 2.13 in terms of $\vec{x}$ and $\vec{\xi}$ results as follows:

$$\mathcal{F}_{1D}[r(\rho,\theta)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mu(x,y)e^{-j2\pi\vec{x}\cdot\vec{\xi}}d\vec{x}. \tag{2.15}$$

To match Equation 2.15 with the definition of the two-dimensional FT, shown through Equation 2.14, the vector $\vec{\xi}$ should represent the two-dimensional frequency variable (ω_x, ω_y). If $\vec{\xi}$ is represented in terms of sines and cosines like:

$$\begin{aligned} \omega_x &= s\cos\theta, \\ \omega_y &= s\sin\theta. \end{aligned} \tag{2.16}$$

An expression matching with the definition of the two-dimensional FT (Equation 2.14) is finally achieved in:

$$\mathcal{F}_{1D}[r(\rho,\theta)] = M(\omega_x,\omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mu(x,y)e^{-j2\pi(x\omega_x+y\omega_y)}dxdy. \tag{2.17}$$

Thus, it is mathematically proven that accordingly to the FST, the one-dimensional FT of the angular projections $r(\rho,\theta)$ describing a two-dimensional distribution $\mu(x,y)$, is equivalent to the two-dimensional FT $M(\omega_x,\omega_y)$ of such distribution.

$$\mu(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} M(\omega_x,\omega_y)e^{j2\pi(x\omega_x+y\omega_y)}d\omega_x d\omega_y, \tag{2.18}$$

it is then possible to recover $\mu(x,y)$ from the inverse two-dimensional FT of $M(\omega_x,\omega_y)$.

$$\mathcal{F}_{1D}[r(\rho,\theta)] = M(\omega_x,\omega_y), \tag{2.19}$$

$$\mathcal{F}_{2D}^{-1}[M(\omega_x,\omega_y)] = \mu(x,y). \tag{2.20}$$

So far, it seems that at least theoretically, the CT inverse problem is solved by means of the FST. Nonetheless, it is compulsory to highlight that the FST proof is formulated by assuming an infinite number of angular projections ρ,θ, that in consequence, can provide a $M(\omega_x,\omega_y)$ function known overall the

Fourier space. In practice, however, there is an implicit limitation in the acquisition of projection data, as it can only be performed in discrete form, so $M(\omega_x, \omega_y)$ is only known over a finite number of slices [1]. Such limitation is the cause of degradation in reconstructed images, and becomes more severe as the acquisition of projection data becomes more restricted. In transmission CT the quality of a reconstruction image depends on the availability of the sampled projection data [90].
Whereas the FST represents a conceptual model of CT image reconstruction, practical implementations require different reconstruction algorithms to address the undesired effects that result from recovering an image only from available projection data.

2.4 Image Reconstruction Methods

Since the appearance of CT, related theory, imaging geometries, reconstruction algorithms, as well as fields of application, have been subjected to a rapid development [18]. Therefore, track down and/or describe every available CT reconstruction method, represents an impossible task.
Section 2.3 proved that mathematically inverting the RT is theoretically exact, but insufficient for a practical implementation, thus different approaches have been developed. The nature of such approaches depend on the application requirements (e.g. high-speed reconstruction, high-detailed reconstructed image) as well as to involved parameters (e.g. acquisition geometry, data availability and quality, computational resources).
In this section, reconstruction methods are broadly classified in terms of the modelling of the inversion problem, into series expansion and transform methods.

2.4.1 Series Expansion Methods

In series expansion methods, the image reconstruction problem is outlined as a system of linear equations [1]. In doing so, the projected object is first assumed as a grid formed by N pixels, with every pixel assigned with a sequential label $\mu_j (j = 1, 2, ...)$. In such grid, the two-dimensional distribution $\mu(x, y)$ is assumed to be constant for every pixel at every j position. In a similar fashion to the pixel labelling μ_j, rays belonging to angular projections, are assigned

with a sequential order identifier $r_i (i = 1, 2, ...)$. This is illustrated in Figure 2.12, where the element denoted by the w_{ij} literal, is the weight of the contribution of the j^{th} pixel μ_j to the i^{th} projection ray r_i. For this example, such contribution refers to the length portion that overlaps a pixel, but other attributes can be used (e.g. attenuation, point spread function) [88, p. 126].

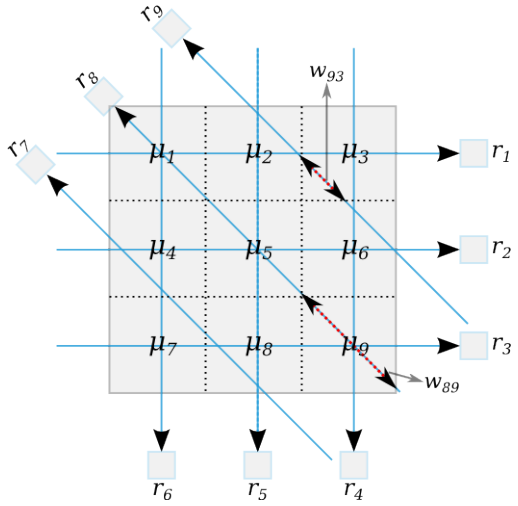


Figure 2.12: Series Expansion Example

As it was previously mentioned in Section 2.1, for the discrete attenuation model shown in Figure 2.5, projection measurements and image pixels are related in terms of ray sums. Therefore, for the example in Figure 2.12, the system of linear equations is obtained through:

$$r_i = \sum_{j=1}^{N} w_{ij}\mu_j, \quad i = 1, 2 \dots M \tag{2.21}$$

$$\begin{cases} w_{11}\mu_1 + w_{12}\mu_2 + w_{13}\mu_3 = r_1, \\ w_{24}\mu_4 + w_{25}\mu_5 + w_{26}\mu_6 = r_2, \\ w_{37}\mu_7 + w_{38}\mu_8 + w_{39}\mu_9 = r_3, \\ w_{43}\mu_3 + w_{46}\mu_6 + w_{49}\mu_9 = r_4, \\ w_{52}\mu_2 + w_{55}\mu_5 + w_{58}\mu_8 = r_5, \\ w_{61}\mu_1 + w_{64}\mu_4 + w_{67}\mu_7 = r_6, \\ w_{78}\mu_8 + w_{77}\mu_7 + w_{74}\mu_4 = r_7, \\ w_{89}\mu_9 + w_{85}\mu_5 + w_{81}\mu_1 = r_8, \\ w_{96}\mu_6 + w_{93}\mu_3 + w_{92}\mu_2 = r_9. \end{cases} \tag{2.22}$$

where M is the total number of rays that take part of the complete projection data set, and N is the total number of pixels in the image grid. If the system of equations is re-written in matrix form, the following expression is obtained:

$$\vec{r} = \mathbf{W}\vec{\mu}, \tag{2.23}$$

with $\vec{r} = [r_1, r_2 \cdots r_9]^T$, $\vec{\mu} = [\mu_1, \mu_2 \cdots \mu_3]^T$, and $\mathbf{W}$ as the weighting matrix (i.e. system coefficient matrix) with M by N dimension. The expansion series method expressed by Equation 2.23, looks simpler than transform-based methods, nonetheless, series expansion methods are known for being limited in speed and accuracy in a certain kind of applications (i.e. medical imaging). Yet, they represent an alternative for situations in which acquiring non-uniformly distributed projections and/or a complete data set, is impossible. Series expansion methods are also preferred to overcome some undesired effects that are implicit in the propagation of energy between the source and detector (e.g. ray bending, attenuation) [1].
From Equation 2.23, it seems that the solution to the inverse problem of finding the object's image, given by $\vec{\mu}$, from the angular projection measurements $\vec{r}$, can be obtained through:

$$\vec{\mu} = \mathbf{W^{-1}}\vec{r}, \tag{2.24}$$

however, direct matrix inversion of $\mathbf{W}$ is not realisable in practice. In a real application, $\mathbf{W}$ is commonly larger than accepted to employ conventional matrix methods, and even computationally impractical when both the number of rays M and pixels N are large [1]. Therefore, in practice, series expansion

methods are used to obtain an approximate set of solutions, rather than an exact solution [87].
As consequence, some practical methods have been developed under the principles of series expansion methods. These methods formulate the reconstruction problem by considering extra information obtained from the employed imaging system (e.g. acquisition geometry, modality, data sparsity), as well as any prior knowledge about the object, to limit solutions to those that are feasible.
Algebraic Reconstruction, Simultaneous Iterative Reconstruction, and Simultaneous Algebraic Reconstruction, are among the most employed series expansion based reconstruction methods [1, 2, 9, 10, 88].

2.4.2 Transform Methods

Transform methods are the counterpart in the category reviewed in this section. For the recovery of the two-dimensional distribution $\mu(x, y)$, from acquired projection data $r(\rho, \theta)$, these kind of reconstruction methods demand the direct utilisation of analytical inversion formulas. Such formulas rely on the theory foundations and models, commonly derived from the FST [2].

2.4.2.1 Direct Fourier Inversion Method

Derived from the FST, the inverse RT represents the most direct formula to recover the two-dimensional density $\mu(x, y)$, through the calculation of the inverse two-dimensional FT of Fourier transformed projection data $M(\omega_x, \omega_y)$, previously shown in Equation 2.18. Although, as previously reviewed during the formulation of the inverse RT in Section 2.3, the application of the inverse RT is not feasible for practical implementations, as it does not allow exact reconstruction [1]. Such limitation arises due to the impossibility to account for the overall Fourier space (ω_x, ω_y), from discrete sampled data $r(\rho, \theta)$. Moreover, for the proper utilization of Equation 2.18, projection data in Fourier domain must be interpolated from the polar $R(\rho, \theta)$ to the Cartesian $M(\omega_x, \omega_y)$, which involve simultaneous calculations of large sets of data [1].

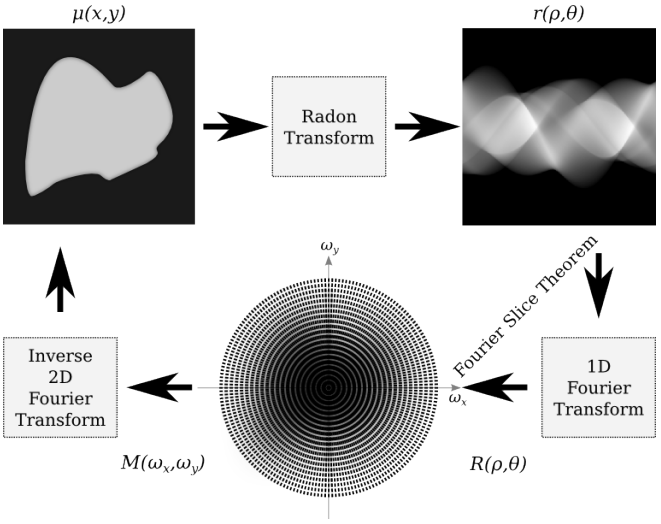


Figure 2.13: Direct Fourier Reconstruction

In Figure 2.13 it can be seen that projection slices ($R(\rho, \theta)$) present an overweight at the origin in the (ω_x, ω_y) plane, which means that the Fourier representation of the object's image $M(\omega_x, \omega_y)$ is not uniformly dense. Such characteristic is a cause of image degradation (i.e. blurring effect) [88]. Nonetheless, depending on the application, if an exact (i.e. closely exact) reconstruction is not critical, but an admissible approximation fulfil requirements, the coordinate system interpolation can be substituted by less demanding alternatives to determine the values on the square grid (e.g. linear interpolation, nearest neighbor). If proper interpolation techniques are accompanied with FFT algorithms, fast reconstruction can be performed [42]. For more information about the mathematics and aspects of direct Fourier methods, [91] is recommended.

2.4.2.2 The Filtered Backprojection Method

In as much as it is proved that the FBP is capable to deliver accurate images at high computational efficiency, it is among the preferred reconstruction methods in CT [11, 12]. Moreover, the FBP was the most important reference reconstruction algorithm in Tomography research for several years, and different generalisations of it, have been developed ever since. The FBP is a transform reconstruction method that has been derived from the inverse RT formulation, aiming to obtain a suitable alternative for practical implementations. The

mathematical explanation about the FBP formula extraction, from the inverse RT formula previously reviewed in Section 2.3, follow the one given in [1, p.63]. By recalling the formula for the recovering of the internal distribution $\mu(x, y)$, by means of the inverse two-dimensional FT,

$$\mu(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} M(\omega_x, \omega_y) e^{j2\pi(x\omega_x + y\omega_y)} d\omega_x d\omega_y, \tag{2.18}$$

it follows that, in order to avoid the polar to Cartesian interpolation limitation of direct Fourier inversion, variables ω_x and ω_y must be translated to an expression in terms of the polar form variables s and θ, to fit with the format in which projection data is acquired. Such conversion can be achieved through the application of the change of variable formula for the double integral case, given by [92]. The formula for a change of variable from (x, y) to (m, n) is shown bellow:

$$\iint_R f(x,y)dxdy = \iint_{R^*} f[x(m,n), y(m,n)] \left|\frac{\partial(x,y)}{\partial(m,n)}\right| dmdn. \tag{2.25}$$

In terms of Equation 2.18 variables, it follows:

$$\begin{aligned} \omega_x &= s\cos\theta, \\ \omega_y &= s\sin\theta. \end{aligned} \tag{2.16}$$

And for differentials $d\omega_x$ and $d\omega_y$:

$$\begin{aligned} d\omega_x d\omega_y &= \left|\frac{\partial(\omega_x, \omega_y)}{\partial(s,\theta)}\right| dsd\theta, \\ &= \left|\begin{matrix} \frac{\partial\omega_x}{\partial s} & \frac{\partial\omega_x}{\partial\theta} \\ \frac{\partial\omega_y}{\partial s} & \frac{\partial\omega_y}{\partial\theta} \end{matrix}\right| dsd\theta, \\ &= \left|\begin{matrix} \frac{\partial(s\cos\theta)}{\partial s} & \frac{\partial(s\cos\theta)}{\partial\theta} \\ \frac{\partial(s\sin\theta)}{\partial s} & \frac{\partial(s\sin\theta)}{\partial\theta} \end{matrix}\right| dsd\theta, \\ &= |s\cos^2\theta + s\sin^2\theta| dsd\theta, \\ &= |s| dsd\theta. \end{aligned} \tag{2.26}$$

Therefore, Equation 2.18 in polar form is given by:

$$\mu(x,y) = \int_0^{2\pi}\int_0^{\infty} M(s\cos\theta, s\sin\theta)e^{j2\pi s(x\cos\theta + y\sin\theta)}|s|dsd\theta. \tag{2.27}$$

By looking at the integration limits in the outer integral of Equation 2.27, it can be realized that if periodicity of the parallel acquisition geometry is taken into account, the following expression is accomplished:

$$M(s\cos(\theta+180°), s\sin(\theta+180°)) = M(-s\cos\theta, -s\sin\theta), \tag{2.28}$$

therefore, Equation 2.27 can be split into:

$$\begin{aligned}\mu(x,y) = &\int_0^{\pi}\int_0^{\infty} M(s\cos\theta, s\sin\theta)e^{j2\pi s(x\cos\theta + y\sin\theta)}|s|dsd\theta \\ &+ \int_0^{\pi}\int_0^{\infty} M(s\cos(\theta+180°), s\sin(\theta+180°))e^{j2\pi s(x\cos(\theta+180°) + y\sin(\theta+180°))}|s|dsd\theta,\end{aligned} \tag{2.29}$$

form which it can be inferred, that the second double integral within the addition of Equation 2.29 is redundant, so it can be discarded. Equation 2.29 then turns to:

$$\mu(x,y) = \int_0^{\pi}\int_{-\infty}^{\infty} M(s\cos\theta, s\sin\theta)|s|e^{j2\pi s(x\cos\theta + y\sin\theta)}dsd\theta. \tag{2.30}$$

By substituting the FT of a projection with respect to s and at a fixed angle θ, $R_\theta(s)$, for the two-dimensional FT $M(s\cos\theta, s\sin\theta)$, Equation 2.30 can be expressed in the following form:

$$\mu(x,y) = \int_0^{\pi}\left[\int_{-\infty}^{\infty} R_\theta(s)|s|e^{j2\pi s(x\cos\theta + y\sin\theta)}ds\right]d\theta. \tag{2.31}$$

Equation 2.31 is the mathematical expression for the FBP, from which the integral within square brackets is the inverse FT of the product between between the projection $R_\theta(s)$ and the term $|s|$. Such product is equivalent to

the filtering, in Fourier domain, between the angular projection $R_\theta(s)$ and the filter frequency response given by $|s|$. The term $|s|$ is also the resultant of the Jacobian of change between Cartesian to polar coordinates given by Equation 2.26.
After calculating the inverse FT of the term inside square brackets, with respect to s, filtered projection at angle θ is written as:

$$\int_{-\infty}^{\infty} R_\theta(s)|s|e^{j2\pi s(x\cos\theta+y\sin\theta)}ds = Q_\theta(x\cos\theta + y\sin\theta), \tag{2.32}$$

If substituting $Q_\theta(x\cos\theta + y\sin\theta)$ for the integral inside square brackets of Equation 2.31, the expression is reduced to a single integral with respect to θ. This integral is responsible for bringing back every line integral value $\rho = x\cos\theta + y\sin\theta$, to its corresponding position over the (x, y) plane, for the recovering of the image $\mu(x, y)$. Such procedure receives the name of backprojection [1, p.65].

$$\mu(x, y) = \int_0^{\pi} Q_\theta(x\cos\theta + y\sin\theta)d\theta. \tag{2.33}$$

The following diagram shows, in a sequential fashion, the steps involved in the computation of the FBP.

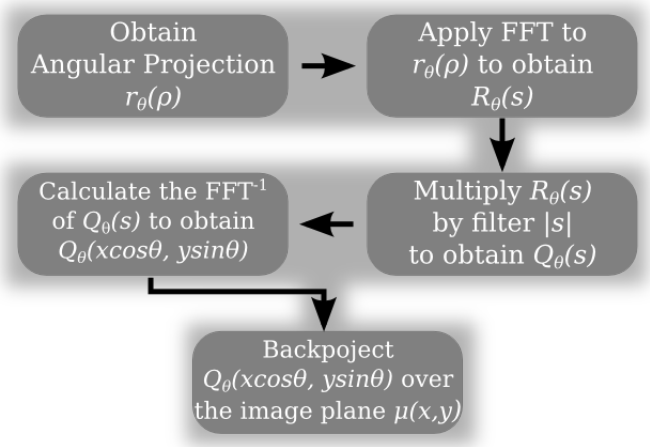


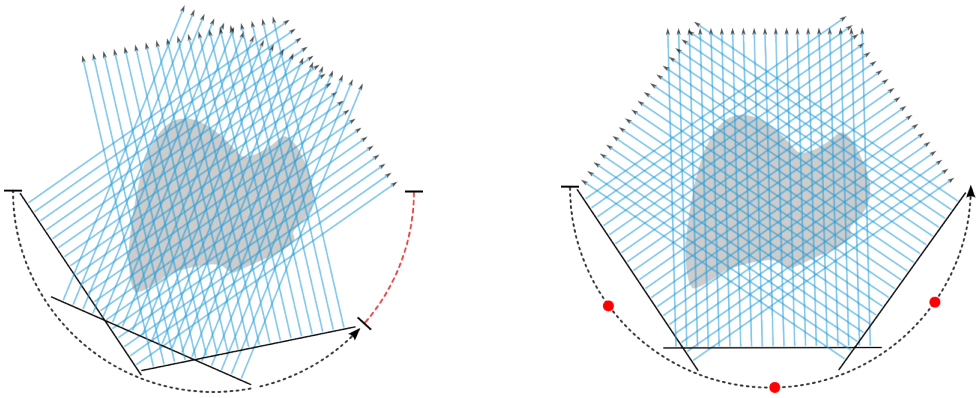
Figure 2.14: The Filtered Backprojection Method

Both elements (Equations 2.32 and 2.33) give the name to the reconstruction method, filtering and backprojection. These elements have a direct implication in the advantages over direct Fourier reconstruction methods. Filtering helps in minimizing the undesired effects caused by the frequency overweight in dc term of the Fourier space, as shown in Figure 2.13. The backprojection reduces the strong dependency of complex approaches needed to perform Fourier domain interpolation, while linear interpolation is commonly enough when carried out in spatial domain [93]. Not least important, is the advantage of the FBP method over the direct Fourier methods to independently compute angular projections, making it possible to initiate that reconstruction process soon after the first angular projection has been acquired [1].
In a comparison between series expansion methods and transform methods, it can be deducted that transform methods perform better in terms of speed and are capable to deliver accurate images, although they are strongly dependent on the vast availability and good quality of projection data. Conversely, series expansion methods are slower and require more computation resources, yet they perform better with limited projection data and are able to incorporate prior knowledge like noise and other physical properties of the imaging system [2, 12].

2.4.2.3 FBP Image Reconstruction from Incomplete Data

Regardless of the disadvantages involved in being a transform method, the FBP is among the preferred reconstruction methods in Tomography and different approaches to overcome its strong dependency on the availability quality projection data, have been proposed.
The effect caused by the non accomplishment to acquire a complete set of projection data, derives in inaccurate image reconstructions, which can be something critical in some kind of applications. The most common sources that drive projection data to incompleteness, are identified to be originated during the acquisition procedure. Such negative sources are broadly classified into two sorts [14].
The first one occurs due to the impossibility to acquire line integral values at a sufficient amount of angle views. Such limitation causes the measured projection data set to be composed by a limited number of angular projections, so it can not afford to provide the invertion method, with a sufficient amount of information, deriving into an inaccurate recontructed image. The following

figure shows two different cases in which the acquisition set-up limits the measurement of a complete projection data set.

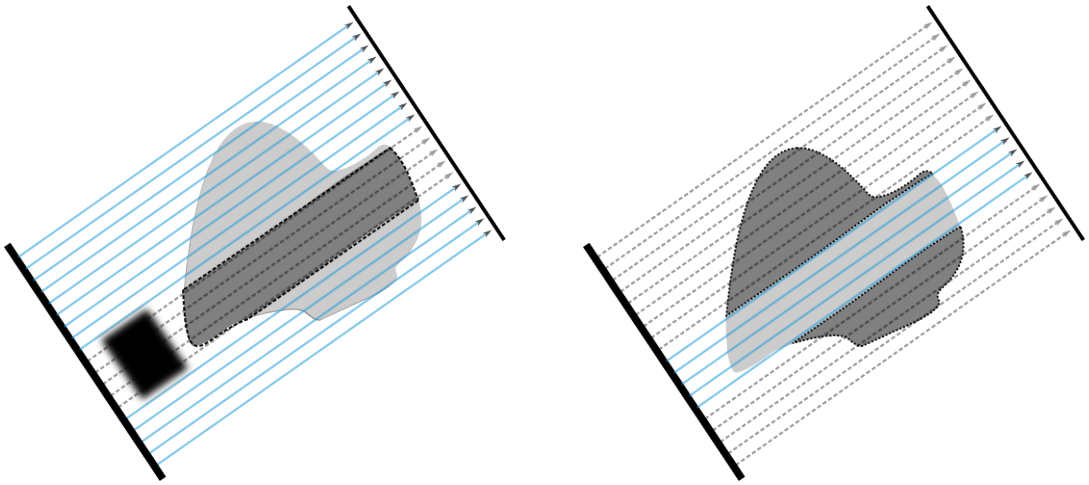


(a) The rotation circumference is limited.

(b) Acquisition is limited to sparse angle views.

Figure 2.15: Limited Angle.

In Figure 2.15a, the angle view is limited at the rotation axis, which can not afford to move the radiating source over the whole angular range, from 0° to 180°. In Figure 2.15b, the radiating source is free to travel along the rotation, although measurements are taken only from a few locations within the angular range, so the projection data set is composed by sparse angular projections. The second source that drive projection data to incompleteness, is when regardless the possibility to measure data over the whole angular range, and from sufficient amount of angular views, some line integrals within the angular projection can not be measured. In this case it is then said that the projection data is truncated. To exemplify the projection data truncation, the following figure is included.



(a) A high absorbing object hinders some of the rays within the projection.

(b) Only few rays within the projection are measured.

Figure 2.16: Truncated Projections.

Two different ways in which projection data is truncated, can be seen through Figure 2.16. In the case shown in 2.16a, truncation is caused by a high absorption object that resides within the Field of View (FOV). Such undesired object blocks the acquisition of line integrals, driving them to either reach the detector with a distorted value, or being absorbed. In Figure 2.16b, even though the FOV is not infringed, some line integrals are not measured. This latter case is of special interest and will be addressed below.
The incomplete data limitation in the FBP method, has been mainly tackled through two different practical approaches [16–19]. The first one deals with estimation of missing data, followed by an appropriate compensation [20]. The second approach consists in the generalisation of the existing methods, to overcome the incomplete data inconsistencies, and accurately reconstruct images from only available data [21–23].
Such second approach inspired the development of reconstruction algorithms, not only to alleviate the undesired effects in reconstruction images that arise from the impossibility to measure a complete set of projection data, but also to achieve the reconstruction of region-specific images by intentionally inducing projection data to incompleteness. That is to say, if data acquisition is consciously limited to data that belongs only to a specific subregion of the entire image (i.e. region of interest), less complex and hence faster, accurate

localised images can be obtained from reduced exposure projection data [24]. It is through the second practical approach, where the WT has been employed as an alternative to alleviate the undesired effects that result due to the incompleteness of projection data. Both the FBP reconstruction method and the WT, were of significant importance in the development of this research. As have been mentioned before in the introductory chapter, the main objective of the inclusion of the WT is to be able to accurately reconstruct localised area images over a ROI to reduce complexity when considering only line integrals passing over.
There are different methods that do not employ the WT but also target the reconstruction of ROIs, some of them being developed at the same time like [14, 21, 22]. According to [18], the state-of-the-art in local/interior Tomography image reconstruction is the Hilbert transform analysis [29, 94, 95]. More recently, the accurate reconstruction of ROIs is being actively developed towards its application in multimodality imaging [19], mainly targeting interior CT-MRI and CT-Single-photon Emission Computed Tomography (SPECT) combinations. The integration of multiple tomographic modalities has been named Omni-Tomography [96], which has been found to have potential in clinical applications. The Omni-Tomography idea has the aim to fuse multiple imaging modalities into a single system in order to be able to extract features that, in many cases, cannot be provided by a single Tomography modality [19].

2.5 Summary

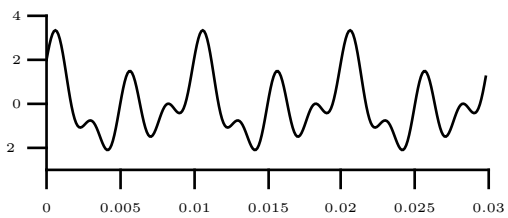
In this research, we deal with transmission tomography, which in its classical form is described by the RT: the forward RT is implemented by transmission measurements, while we assign ourselves the task to find efficient solutions for the implementation of the inverse RT, and reconstruct the imaged object. This is strictly correct only to the extent in which transmission measurements can yield correct attenuation line integrals: this is obviously questionable if the object manifests strong diffuse scattering and the attenuation values are not possible to be calculated accurately, just from the transmission values.
The RT treatment in this chapter refers to parallel beams, but it is applicable also to other geometries (e.g. fan beams), by applying a few mathematical manipulations. It is also worth noting that in Chapter 4 the forward RT model

will be used independently of the acquisition procedure: it will allow the calculation of the block-tile supports and optimise the image reconstruction. The mathematical formalism of the RT and FST is developed in polar coordinates, which allows to exploit the filtering concept. The Jacobian of coordinate change appears in the inverse RT integral, which can now be interpreted as convolution between the object and a filter, since it contains the product between two Fourier images. This gives rise to the argument that a variety of filtering options can be deployed to account for constraints and specific object characteristics. The FST is a popular means to estimate the inverse RT. However, since in reality all forward RT datasets represent only a sample of the RT, the FST delivers reconstruction quality which is a function of the sampling parameters.
We take advantage of an essential aspect of the FST, namely that it can be applied as a mathematical formalism to a variety of objects, including their possible partial representations, such as wavelet and scaling coefficients. This will be exploited in Chapter 4

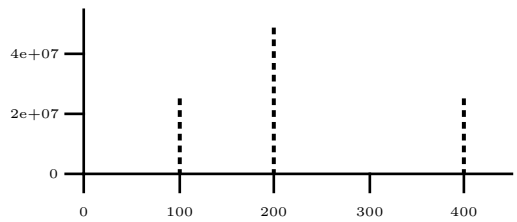
Chapter 3

The Wavelet Transform

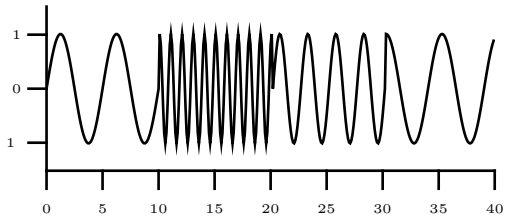
The WT can be thought as an extension of the FT, which can be employed for the analysis of non-stationary signals (Figure 3.1). The main characteristic of a non-stationary signal is the presence of abrupt frequency changes along its lifespan, feature for which the FT results to be inadequate. Non-stationary signals demand information not only about the frequency components of a signal, but also about the time at which these components are emitted. When applying FT to a signal, time information is not lost, but deeply hidden within the phases shifts of sines and cosines that represent different moments of the signal. The phases shifts of sine and cosines cause components to amplify or cancel each other, making it impossible to extract accurate time information.



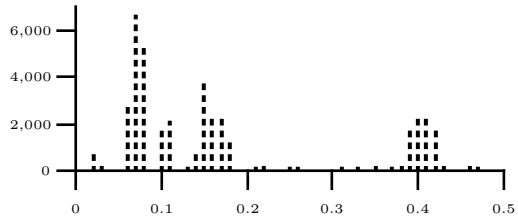
(a) Stationary Signal.



(b) Stationary Frequency Components.



(c) Non-Stationary Signal.



(d) Non-Stationary Frequency Components

Figure 3.1: The Inefficiency of the FT to represent Non-Stationary Signals.

3.1 The Short-Time Fourier Transform

A precedent in the realisation of the WT was the practical approach proposed by Dennis Gabor in 1946 [97]. In the literature, Gabor's model is given different names: Short-Time Fourier Transform (STFT), Windowed Fourier Transform (WFT) and Gabor Transform (GT). In order to review the STFT, it is important to highlight first the mathematical expression for the FT through Equation 3.1, in which a signal under analysis $f(t)$, is entirely correlated with an infinite number of sines and cosines at different frequencies.

$$F(\omega) = \mathcal{F}[f(t)] = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt. \tag{3.1}$$

Conversely to the FT, in the STFT the signal $f(t)$ is split into segments of time before being correlated with small oscillating functions, at different frequencies [98]. The size of the time segments, as well as the support of the oscillating functions, are defined through a fixed-size window function $w(t)$.

$$F(\omega, \tau) = \mathcal{STF}[f(t)] = \int_{t} [f(t) \cdot w(\tau - t)]e^{-i\omega t}dt. \tag{3.2}$$

The window function $w(t)$ is then shifted to analyse the original signal segment by segment, and hence keep record of involved frequency components happening at certain intervals of time.

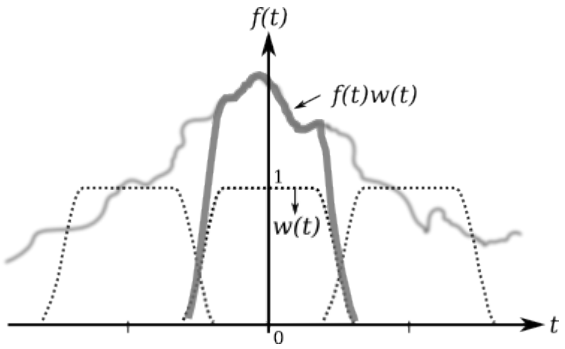


Figure 3.2: The Short-Time Fourier Transform [99].

Although the STFT is able to provide time-frequency information about a signal, the fixed size of the $w(t)$ imposes a compromise that makes the method

suitable only for some cases. If the width of the window $w(t)$ is defined to be very narrow, high frequency events within the analysed signal will be located; however, the window function will not be wide enough to register low frequency portions of the signal. In this case, the analysis data will provide have a good time resolution, but poor frequency resolution. This is shown below in Figure 3.3.

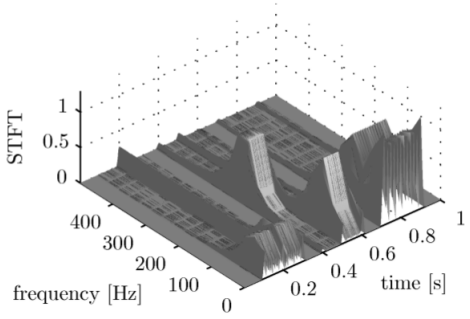


Figure 3.3: Small $w(t)$, Good Time Resolution [100].

In the opposite case, if the window function $w(t)$ is chosen to be wide, time resolution will not be sharp enough to locate brief change events, but a wider frequency spectre will be covered. In this case, analysis data will be able to provide a good frequency resolution, but will result with a poor time resolution. This is shown below in Figure 3.4.

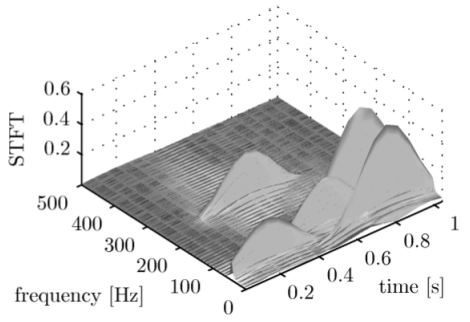


Figure 3.4: Wide $w(t)$, Good Frequency Resolution [100].

Therefore, due to the time-frequency resolution compromise shown in the STFT, the WT arose as the alternative predilection, for the analysis of non-stationary signals.

3.2 The Continuous Wavelet Transform

Similar ideas to the WT date back from as early as the beginning of the last century, when the mathematician Alfréd Haar constructed the Haar orthonormal basis [101]. Years after, again in the Mathematics field, Littlewood and Paley developed a wavelet-like framework applied to the study of trigonometric series [102]. Physics was not an exception and the re-normalisation work developed by Kenneth Wilson [103] is also considered part of the WT pecedent. In more recent times, within the field of engineering, David Marr exposed similar ideas in Robotics [104]; Crosier, Esteban and Galand, as well as Mintzer, did the same in Digital Signal Processing [105, 106]. Despite the previously mentioned pioneering work, the realization of wavelets as a unified framework is attributed to Jean Morlet and Alex Grossmann [38, 107]. Jean Morlet, a geophysicist working in oil industry, informally developed wavelets while looking for an alternative tool to be applied in oil prospecting. In the 1960s the standard technique employed in the search of underground oil consisted in sending vibrations to produce echo from the underground. Such echo were analysed with the aim of gathering information about the composition, deepness and thickness of underground layers.
Analysis of the frequency components of seismic reflected signals, were related to the thickness of the various underground layers; high frequency components corresponded to thin layers, while lower frequencies corresponded to wider layers (Figure 3.5) [108].

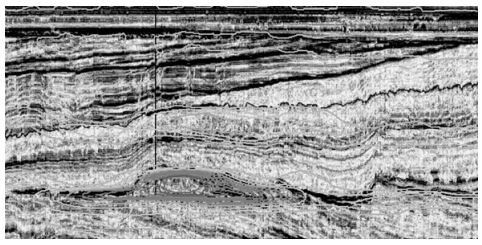


Figure 3.5: Seismic Reflection Signals *"tangle"* [109].

According to the description of seismic body waves, a reflected signal is able to bring together information from as many layers as in the order of hundreds, where each layer corresponds to a certain frequency component interfering with each other; this kind of signals can be clearly categorised as non-stationary (Figure 3.1c).

In the beginning, the FT was the method used to extract information related to every underground layer, but later when computers became more accessible, the STFT started to be employed. In [38, p. 26], Morlet describes that, even when the computer technology at that time allowed intensive STFT analysis, he never managed to access information about layers of different thicknesses. The main cause was the STFT resolution limitation derived from the fixed size of the window function, but moreover, Morlet also ran into the drawback that the STFT gave no numerical way to be synthesized. To work out the STFT resolution limitation, Morlet experimented in stretching and compressing the width of the window function, while keeping the number of oscillations inside constant. The experiment proved that, as opposed to the STFT (Figure 3.6), modifying the width of the window induced a change in the frequency of oscillations, but not the same for its shape. Because no matter if the window function was stretched or compressed, the shape of the oscillations inward remained nearly the same (Figure 3.7), Morlet called such kind of functions "wavelets of constant shape" [110, 111].

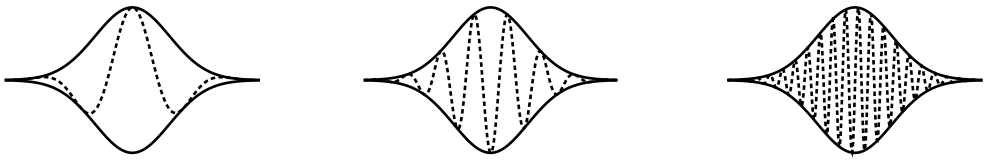
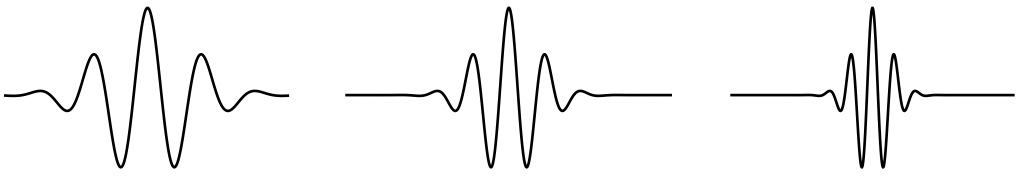


Figure 3.6: STFT Windowing.



(a) Stretched Wavelet. (b) Mother Wavelet. (c) Compressed Wavelet.

Figure 3.7: Wavelet Functions.

In the endeavour to convince his colleagues that his newly developed method was a worthwhile mathematical tool, Morlet teamed up Alex Grossmann, a

theoretical physicist who was referred from one of his student days' friends [112]. Together, Morlet and Grossmann, mathematically validated the empirical results previously obtained. Grossmann then constructed an inversion formula for Morlet's method, and for the first time introduced the term *wavelet* in [113]. In the Continuous Wavelet Transform (CWT), wavelet functions are derived from a main function called *mother wavelet*, whose width and position are varied in order to obtain scaled and translated versions of it. Such *mother wavelet* act as a template to obtain wavelet functions, that may fit better in the analysis of frequency components along the lifespan of a signal. In order to cover different frequency components, the scale s is assigned with different values; $s > 1$ results in the dilation of the wavelet, whilst $s < 1$ compresses it (e.g. Figure 3.7). Wavelet functions $\psi_{\tau,s}(t)$ with translation τ and scale s, are derived from the *mother wavelet* ψ through the following expression:

$$\psi_{\tau,s}(t) = \frac{1}{\sqrt{|s|}}\psi\left(\frac{t-\tau}{s}\right), \quad s, \tau \in \mathbb{R}, \quad s \neq 0. \tag{3.3}$$

The CWT analysis consists in the convolution between a signal $x(t)$ and the set of wavelet basis functions $\psi_{\tau,s}$, to turn the single variable dependent signal $x(t)$ into a function $W_{x;\psi}(\tau, s)$ that depend on two variables, scale and translation (*i*.e. time location). Therefore, through the CWT analysis, wavelet basis functions at scales s, are shifted along the signal lifespan whilst varying the translation parameter τ; this way the time-frequency representation $W_{x;\psi}(\tau, s)$ is built. The following expression is the CWT $W_{x;\psi}(\tau, s)$ of the signal $x(t)$.

$$W_{x;\psi}(\tau, s) = \int_{-\infty}^{\infty} x(t)\frac{1}{\sqrt{|s|}}\psi^*\left(\frac{t-\tau}{s}\right)dt. \tag{3.4}$$

A way to understand the role of scale s, is to imagine it as the inverse of frequency, thus for a wide scale s, global perception of the signal is obtained (i.e. low frequencies), and conversely, a narrow scale s is used to get into more detailed information of the signal (i.e. high frequencies).
In [113], Morlet and Grossmann set a precedent in the field by not only managing to create a method capable to analyse a signal from its coarse components to finer and finer details, but also to recover the original signal back. In effect, the CWT is an invertible transform, which is fulfilled by the

formula shown below (Equation 3.5). It is important to mention that because of the high redundancy of the CWT (i.e. a one-dimensional signal is mapped into a function that depends on two variables), different synthesis formulas are available. Equation 3.5 is the reconstruction formula for the case in which wavelet functions ψ are real-valued [114]. In the formula below, it is possible to see that the integral corresponding to the scaling parameter s, is restricted to consider only positive values.

$$x(t) = \frac{2}{C_\psi} \int_0^\infty \left[\int_{-\infty}^{\infty} W_x(\tau, s) \psi_{\tau,s}(t) d\tau \right] \frac{ds}{s^2}. \tag{3.5}$$

The C_ψ term in the inverse CWT is the *admissibility constant*, which is the result of an imposed requirement to any $\psi \in L^2(\mathbb{R})$ function, that if fulfilled can be considered as a mother wavelet [115]. Such requirement is referred to as the *admissibility condition*, shown below:

$$0 < C_\psi = \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|}{|\omega|} d\omega < \infty, \tag{3.6}$$

where Ψ is the FT of function ψ. The importance of the *admissibility condition* rely in that its fulfilment, assure that the energy of the analysed signal $x(t)$ is preserved by the CWT [115]. However, to achieve perfect reconstruction, the transform is required to analyse a signal at all possible resolutions and displace the wavelet functions by all values. Such a characteristic showed that the method was not yet suitable for practical applications, its computation was slow and painful, and it was necessary to study the transformed signal at intermediate resolutions; the process was redundant.
Although, such redundancy can be exploited in certain cases, as it makes a lot easier to interpret results and outline conclusions from collected data. Redundant data is useless for applications that require to compress data, but useful for applications in which the main objective is to find patterns within hidden information. Redundancy is also known to give more freedom in choosing the wavelet function to be employed in the CWT analysis [114].

3.3 The Discrete Wavelet Transform

Yves Meyer, who was a mathematician immersed in the mathematics of wavelets, had the belief that redundancy was something unavoidable in the task of obtaining a sufficiently local time-frequency representation of a signal. To prove his conviction, Meyer worked in demonstrating that orthogonal wavelets did not exist, but he failed. Contrary to his hypothesis, and for the benefit of the study field, Meyer ended up constructing the orthogonal wavelet function that he previously denied [116]. Meyer's discovery implied that decomposing a signal in an orthogonal wavelet basis, provided a representation with as many points as the signal itself, so redundancy could be avoided.
Although orthogonal wavelets are much more difficult to construct, they are considerable more liable to achieve fast algorithms [38, p. 125]. Orthogonal wavelets are a special case of discrete wavelet functions, whose scale and translation parameters s and τ, receive discrete values that differ by a power of two (i.e. dyadic).

$$\psi_{i,k}(t) = \frac{1}{\sqrt{s_0^k}}\psi\left(\frac{t - i\tau_0}{s_0^k}\right), \quad \text{with } i \text{ and } k \text{ dyadic,} \tag{3.7}$$

where s_0 and τ_0 are initial scale and translation, which give the starting point for the transform analysis, as well as make it possible to speed up the calculation of wavelet coefficients W. These kind of wavelet functions turns the CWT to a different transform framework, the Discrete Wavelet Transform (DWT) [117].

$$W_{x;\psi}(\tau_0^i, s_0^k) = \int_{-\infty}^{\infty} x(t)\psi_{i,k}^*(t)dt. \tag{3.8}$$

Orthogonal wavelets have also a direct implication in the synthesis of the DWT when avoiding redundancy in wavelet signal analysis. By employing orthogonal basis functions, the original signal can be recovered simply by accumulating the scalar products between wavelet coefficients and wavelet basis functions.

$$x(t) = \sum_{\tau_0^i}\sum_{s_0^k} W_x(\tau_0^i, s_0^k)\psi_{i,k}(t). \tag{3.9}$$

The work initiated by Morlet, mathematically validated by Grossmann and

improved by Meyer, turned into an easy to compute discrete orthogonal transform, considered concise and that allowed perfect reconstruction.

3.4 The Fast Wavelet Transform

As previously reviewed, the pioneering work developed by Morlet and Grossmann set the precedent in the development of the WT. A few years later, Meyer improved such foundations by creating an orthogonal wavelet function, and managed to drive the WT towards a practical implementation. Around the middle of the 1980s, under a different perspective, Stéphane Mallat proposed a different framework for the computation of the DWT; the FWT [37].
Mallat envisioned that what mathematicians were doing with wavelets at that time was the same thing that engineers and researchers in the fields of signal and image processing were doing, but under different names. The most clear example was that multiresolution processing of images was already used to obtain a better representation of information within images [118–121]. More specifically in pattern recognition, where the analysis of signals was performed by separating information in a hierarchical way; from a coarse resolution (i.e. larger structures that define the image context) to gradually finer resolutions (i.e. containing fine details of the image) [118, 122].
The pyramidal hierarchical model of Burt and Adelson, and Crowley, for the computation of images at different resolutions [123, 124], as well as the Quadrature Mirror Filtering (QMF) of Esteban and Galand, for speech subband coding [125], attracted Mallat's interest in the development of the multiresolution theory presented in [37], and that gave the necessary support for the creation of the FWT.
In Burt and Adelson's model, the computation of multiresolution details is simplified by imposing a dyadic resolution step. At every resolution level 2^j, details are obtained through the filtering of the original image (i.e. base image). Such filtering consists in the difference of two low-pass filters, which is followed by a subsampling by a factor of 2^j. The filtering and subsampling operations are dependent on the desired finite resolution. The resulting details, which represent data at different resolution levels j, are arranged into a pyramid structure called *Laplacian Pyramid* (Figure 3.8) [123].

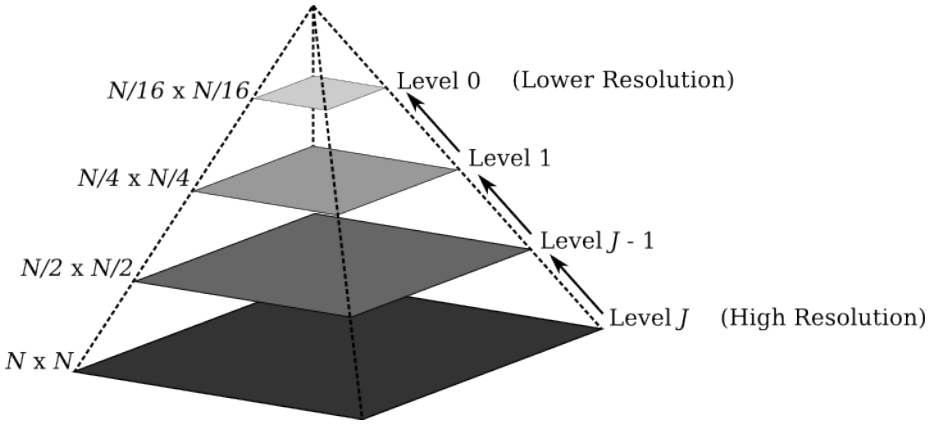


Figure 3.8: Pyramidal Image Structure.

In [37], Mallat reviewed the pyramidal algorithm and identified two main difficulties in its implementation: the first one is related to the correlation between resolution levels in the pyramidal separated data. Such correlation lacks of a clear model to deal with, so it is difficult to identify if similarities between image details at different resolution steps, are a consequence of the image properties or arise from the representation redundancy. The second difficulty that Mallat found, was that the algorithm does not allow any spatial orientation selectivity during the decomposition process. This difficulty was interpreted by Mallat as spatial homogeneity, which is a limitation for pattern recognition applications.

The QMF model of Esteban and Galand was developed with the aim of improving the signal to noise ratio of speech signals for digital telephony. More specifically, in [125], the authors propose a model to avoid the aliasing effects that derive from the sampling decimation when a signal is divided into subbands, during speech processing. Such model consists of an arrangement of mirror half-band low-pass filters, associated with decimation/interpolation techniques (i.e. subsampling and upsampling).

Figure 3.9 shows the QMF, where H_1 is a half-band low-pas filter and H_2 its corresponding half-band mirror filter. Such filters split the sampled input signal $x(n)$ into low and high half-bands $x_1(n)$ and $x_2(n)$ respectively. As the new frequency of each component is half the bandwidth of the original signal, they can be subsampled by means of the Nyquist theorem, by keeping one sample out of two. Output signals are denoted by $y_1(n)$ and $y_2(n)$.

The inverse process is performed by interpolating zero values in-between samples of signals $y_1(n)$ and $y_2(n)$, and filtering by K_1 and K_2. The resulting signals $t_1(n)$ and $t_2(n)$ are finally converged into the output signal $s(n)$.

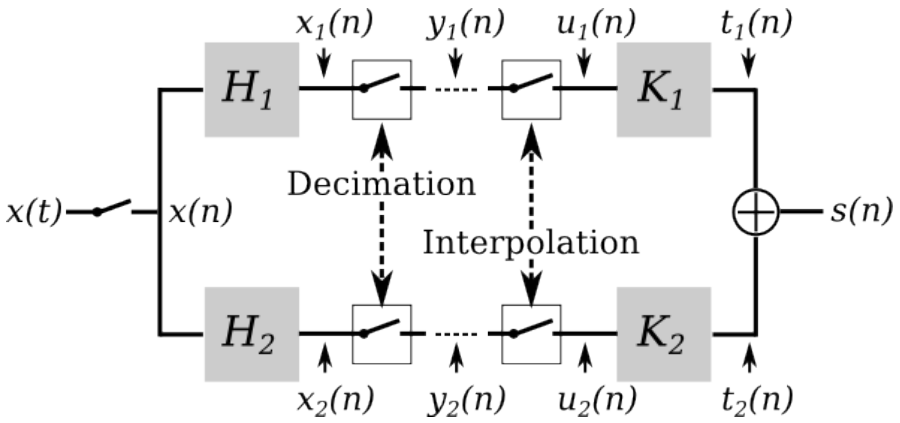


Figure 3.9: Subband Filter Bank.

In the proposed QMF, Esteban and Galand not only managed to improve the signal to noise ratio of subband coded images, but also demonstrated that their model was able to reconstruct quasi-exact signals without using sophisticated filters [125].
Mallat's multiresolution theory managed to unify the three reviewed approaches [126]; the filtering techniques of the QMF applied to speech processing [125], the pyramidal image processing [123, 124], as well as the orthogonal wavelet basis discovered by Meyer [116].
The FWT resembles to the subband filter bank for QMF shown in Figure 3.9, in the sense in which, using wavelets to analyse a signal at different resolutions, can be viewed as a repetitive action of filtering the signal, whilst using wide wavelets to remove everything, but low frequency components, and using narrow wavelets to filter out everything, but high frequency components [38, 127].
High-pass filtering is associated with the wavelet function and low-pass filtering is associated with the scaling function. This is illustrated in Figure 3.10.

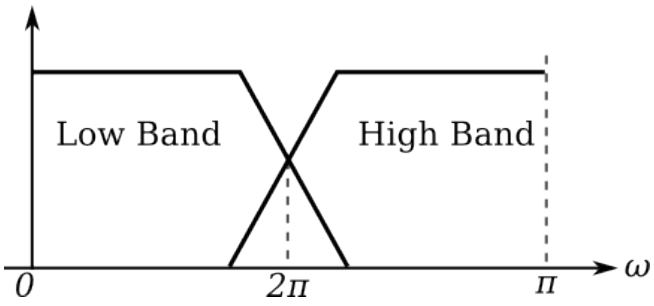


Figure 3.10: Subband Frequency Decomposition.

The influence of Burt and Adelson's pyramidal algorithm in the FWT, can be noticed in the role that the scaling function (low-pass filtering) has for the analysis of a signal at different resolutions. Although, in the pyramidal model, the original signal is always the input of the algorithm, no matter what the desired resolution level is. Conversely, in the FWT analysis, the original signal is the input only for the initial resolution decomposition step (i.e. finer resolution), and coarser resolutions are obtained iteratively by taking the scaling function output (i.e. low-pass filter output), at each level, as the input for the next iteration. In the FWT, resolution levels are therefore computed from fine to coarse. As expressed in [38, p. 154], in the language of wavelets, one can think of the scaling function as the responsible to obtain a version of the input signal at half resolution; in signal processing language, the input signal is low-pass filtered and the result subsampled.
The first step in the analysis FWT consists in decomposing an input discrete signal into two parts: a signal containing the low frequency components (approximations), and a signal containing high frequency components (details). Low frequency components are obtained through the convolution between the input signal $x(n)$ and low-pass filter $h_s(n)$, and high frequency components through the convolution between $x(n)$ and the high-pass filter $h_\psi(n)$. After filtering, the bandwidth of the resulting signals, in comparison with the original signal, is reduced by half. By means of Nyquist theorem, such signals can be represented without any loss, with as half as many samples. Filtered signals are therefore subsampled by a factor of two.

$$W_s(k) = h_s(n) * x(n) \downarrow_{n=2k, \quad k\geq 0} . \tag{3.10}$$

$$W_{\psi}(k) = h_{\psi}(n) * x(n) \downarrow_{n=2k, \quad k \geq 0} . \tag{3.11}$$

Filtered and subsampled signals $W_s(k)$ and $W_{\psi}(k)$, are approximations and details wavelet coefficients. Approximations is the name given to the lower frequency signal, which is a somewhat smooth version of the original signal. The high frequency part of the wavelet decomposition is known as details, and is the signal associated with the finishing touches and/or fluctuations that would have to be added to the approximations to recover the original signal. Approximations and details are separated through the analysis filter bank shown in Figure 3.11.

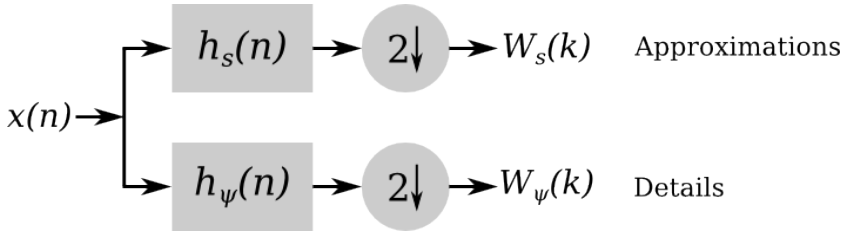


Figure 3.11: FWT Analysis Filter Bank.

In the second step, wavelet coefficients $W_{\psi_1}(k)$ that encode the calculated details (i.e. high frequencies), are saved and the procedure of filtering and subsampling is repeated, but with the approximations $W_{s_1(k)}$ as the new input. Approximations are then divided into two more parts: new approximations $W_{s_2(k)}$ representing even lower frequencies, and new details $W_{\psi_2(k)}$ twice as big as the previously obtained. Both $W_{s_2(k)}$ and $W_{\psi_2(k)}$ are a quarter the resolution of the original signal $x(n)$. Figure 3.12 shows the filters and subsamplers arrangement for a two level decomposition analysis.

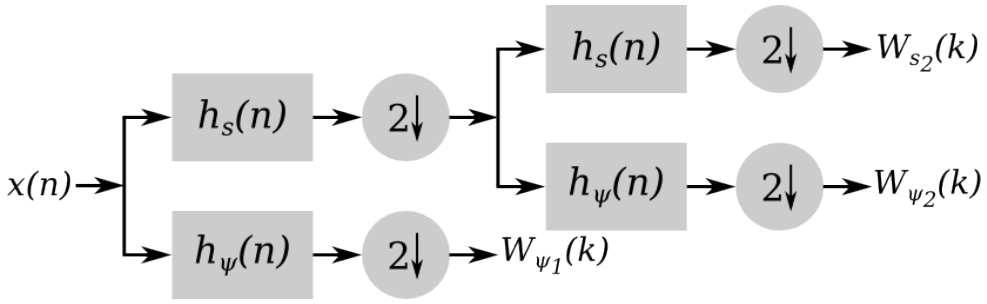
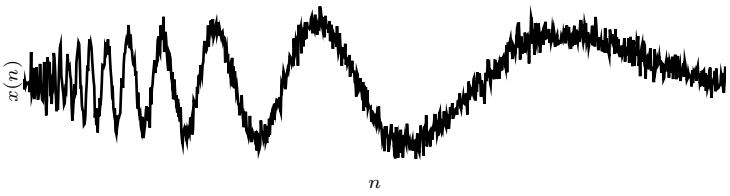
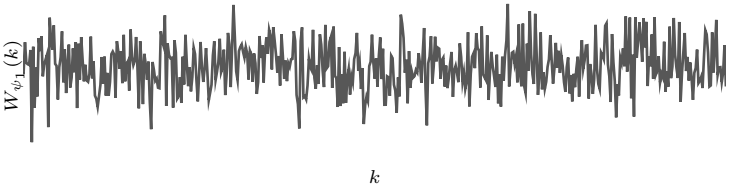


Figure 3.12: FWT Analysis Filter Bank for a Two-Level Resolution Decomposition.

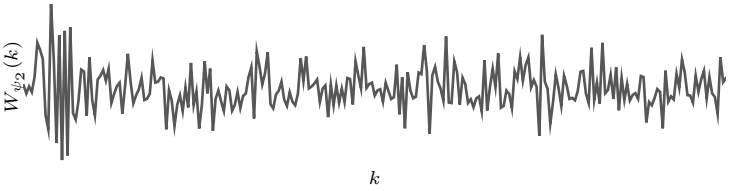
An example showing the decomposition of a real signal is shown below. The top signal is the original signal $x(n)$, which is the input for a two-level resolution FWT analysis, as the one shown in Figure 3.12. Signals shown in Figures 3.13b and 3.13c are the resulting wavelet coefficients (i.e. details). The signal shown in Figure 3.13d contain all the remaining low frequencies (i.e. approximations).



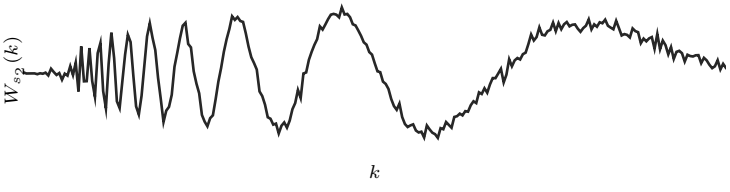
(a) Input Signal $x(n)$.



(b) Details $W_{\psi_1}(k)$ at Resolution Step 1.



(c) Details $W_{\psi_2}(k)$ at Resolution Step 2.



(d) Remaining Approximations $W_{s_2}(k)$.

Figure 3.13: Wavelet Analysis Example for a Two-Level Resolution Decomposition.

Analysis iterations are performed based on the required resolution levels, or up to eventually smooth the original signal out of existence. For the latter case, all the information will be encoded by the wavelet coefficients (details), which will be classified in terms of its corresponding resolution step. If analysis is stopped before the signal dies out, all the remaining information will be encoded by the scaling coefficient (i.e. approximations). Every iteration involve half as many signal samples (Figure 3.14), so the FWT analysis computation becomes twice as fast at every resolution step.

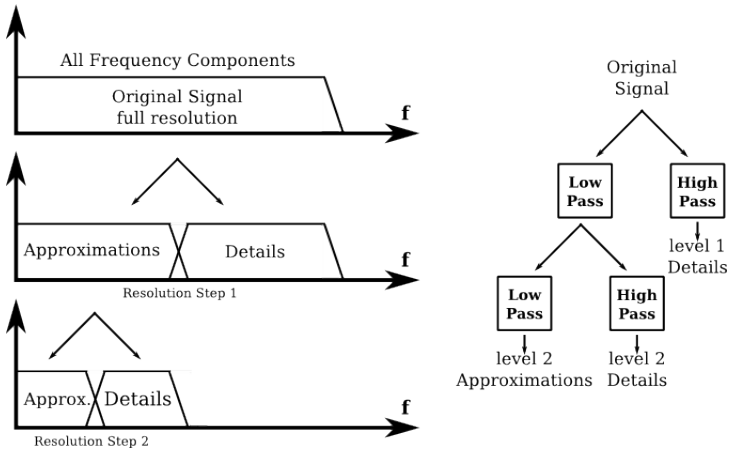


Figure 3.14: FWT Multiresolution Frequency Decomposition.

As it must be expected, the FWT is as well an invertible transform. The synthesis of signal $x(n)$ from its corresponding details $W_{\psi}(k)$, and approximations $W_s(k)$ can be formulated as the inverse FWT. In such formulation, filters are time reversed versions of the analysis counterpart, and subsamplers turn to upsamplers.

$$W_{s_{j-1}}(k) = h_s(-n) * W_{s_j}(k) \uparrow^{2k} . \tag{3.12}$$

$$W_{\psi_{j-1}}(k) = h_{\psi}(-n) * W_{\psi_j}(k) \uparrow^{2k} . \tag{3.13}$$

Equations 3.12 and 3.13 resemble to the synthesis part of the filter bank previously shown in Figure 3.9, where filtered signals are reverse filtered and converged to recover the original signal.

$$x(n) = h_s(-n) * W_{s_j}(k) \uparrow^{2k} \quad + \quad h_{\psi}(-n) * W_{\psi_j}(k) \uparrow^{2k} . \tag{3.14}$$

The calculation of the inverse FWT begins with the upsampling of the details and approximations coefficients $W_{\psi_j}(k)$ and $W_{s_j}(k)$ to yield signals with double the number of samples as its nearer upper resolution. Upsampled signals are then convolved with the reverse filters $h_\psi(-n)$ and $h_s(-n)$, and the resulting signals added to obtain $x(n)$. This is illustrated below in Figure 3.15.

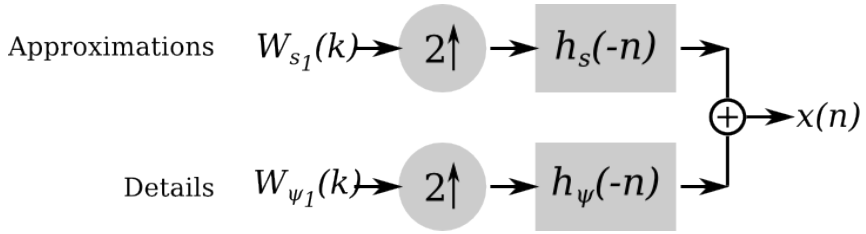


Figure 3.15: FWT Synthesis Filter Bank.

For the inverse FWT computation from multiresolution coefficients, the process is again forward counterpart to the multiresolution analysis (Figure 3.16). Approximations and detail coefficients, at a certain resolution, are upsampled and reverse filtered to yield the approximations coefficient at one step higher in resolution. Such a procedure is iteratively repeated up to recover the original signal, which is at the higher resolution.

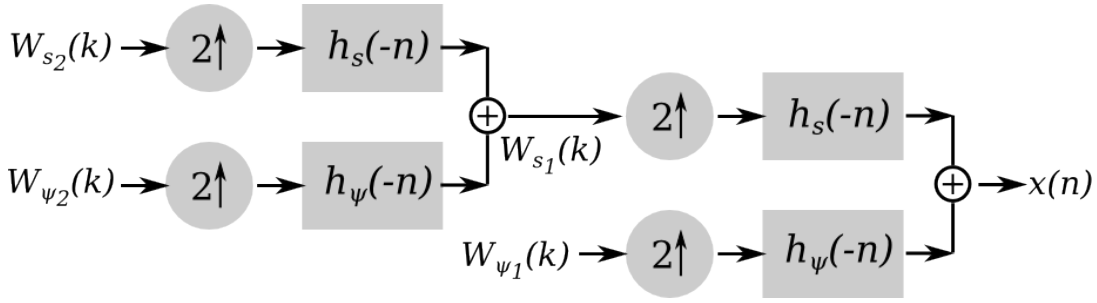


Figure 3.16: FWT Synthesis Filter Bank for a Two-Level Resolution Decomposition.

It is important to mention that the existence of the FWT depends on the availability of a scaling function, as well as the orthogonality of both basis functions, scaling and wavelets. An alternative case are the biorthogonal basis functions, which synthesis functions are not time reversed versions of the analysis filters, but can also provide exact reconstruction, just as the orthogonal basis functions [127].

As previously mentioned along this review, Mallat manage to merge concepts that were typical of a different framework with the Meyer's DWT scheme. At that time, Mallat employed filters that belonged to Meyer and Battle-Lemarié orthogonal basis, which were infinite functions that he truncated to achieve direct implementation into its FWT scheme [112]. In order to avoid the undesired effects caused by the function's truncation, Daubechies proposed a new form of orthogonal finite wavelet basis functions with compact support, as the ones previously shown in Figure 3.7 [128]. The addition of Daubechies' compactly supported wavelets to the FWT made it possible to obtain a time-frequency representation of a signal with ease and accuracy.
The time and frequency information are usually considered as inverse domains, when representing signals (i.e. functions). By means of the Uncertainty Principle [38], it is not possible to simultaneously obtain information about time and frequencies components of a signal, although compactly supported (well-localised) functions like the ones proposed by Daubechies, made it possible to analyse signals at different resolutions, either in frequency or time. Figure 3.17 shows the time-frequency tiles for a delta function for wavelet orthogonal basis, where each tile represent equally portions of the time-frequency plane [127].

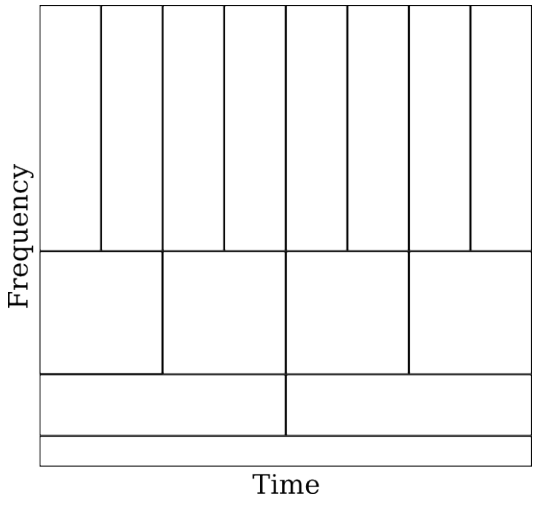


Figure 3.17: FWT Time-Frequency Plane.

Daubechies wavelet basis adapt automatically to signal components, wide wavelets perform better in localising frequency and very narrow wavelets perform better in localising time. Such behaviour of wavelets can be seen by

looking at the time-frequency plane in Figure 3.17: at high frequencies, the width of tiles is smaller, which means that the time resolution is better. Conversely, at low frequencies, the tiles' width gets bigger, so frequency resolution is better [127, p.386]. Although the time-frequency resolution compromise may look as a limitation, the WT has been proved to perform perfectly for a wide range of applications.

3.5 The 2D Fast Wavelet Transform

The two-dimensional FWT is a direct generalisation of the one-dimensional scheme, previously reviewed in Section 3.4. In this two-dimensional version, three wavelet functions and one scaling function are required. Wavelet functions are obtained through separable products between the scaling and wavelet functions s and ψ [129, p.244]. Wavelet functions $\psi(n, m)$, are used to measure high frequency variations along different directions: horizontal, vertical and diagonal.

$$\begin{aligned} \psi^H(n,m) &= s(n)\psi(m) \quad \text{Horizontal.} \\ \psi^V(n,m) &= \psi(n)s(m) \quad \text{Vertical.} \\ \psi^D(n,m) &= \psi(n)\psi(m) \quad \text{Diagonal.} \end{aligned} \tag{3.15}$$

The scaling function $s(n, m)$ is obtained through the separable product between scaling functions along x and y axis.

$$s(n,m) = s(n)s(m). \tag{3.16}$$

In the two-dimensional FWT, the analysis filter bank is built by using three one-dimensional mirror filters, which configuration match with the separable products for each of the wavelet and scaling functions, and subsamplers either relevant across rows or columns. The output of the two-dimensional FWT, consists of three details and one approximations coefficients. Each one of these four different output coefficients, present different enhanced characteristics. Figure 3.18 shows the analysis filter bank for the two-dimensional FWT.
In the two-dimensional FWT filter bank, output wavelet coefficients $W_\Psi^H(m, n)$, $W_\Psi^V(m, n)$, and $W_\Psi^D(m, n)$, refer to the high frequency components (i.e. details) along horizontal, vertical and diagonal directions. The scaling output coefficient

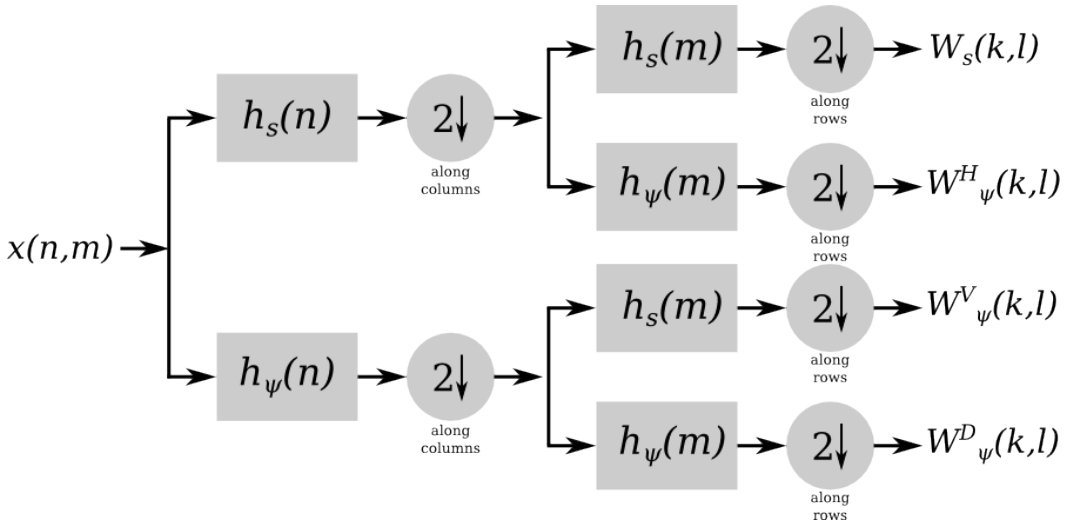


Figure 3.18: Two-Dimensional FWT Analysis Filter Bank.

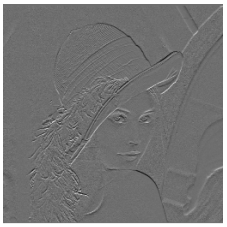
$W_s(m,n)$ corresponds to the low frequency components (i.e. approximations), which are present overall the two-dimensional signal.

$$\begin{aligned} W_s(k,l) &= h_s(m) * [x(n,m) * h_s(n) \downarrow_{n=2k}] \downarrow_{m=2k}, \\ W_\psi^H(k,l) &= h_\psi(m) * [x(n,m) * h_s(n) \downarrow_{n=2k}] \downarrow_{m=2k}, \\ W_\psi^V(k,l) &= h_s(m) * [x(n,m) * h_\psi(n) \downarrow_{n=2k}] \downarrow_{m=2k}, \\ W_\psi^D(k,l) &= h_\psi(m) * [x(n,m) * h_\psi(n) \downarrow_{n=2k}] \downarrow_{m=2k} . \end{aligned} \tag{3.17}$$

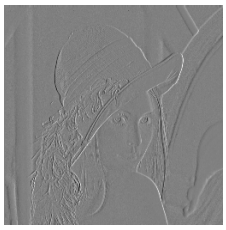
An example of a two-dimensional FWT analysis applied to *lena*, is shown in Figure 3.19 to visualise the characteristics enhanced by every output coefficient. The approximations coefficient image displays very similar information to the input image, which means that the low frequency components of the input signal, contain most of the information. The detail coefficients, represent the high frequency components that can be seen as traces in every different direction. As in the two-dimensional FWT, output coefficients are subsampled by a factor of 2, so every coefficient is half the size of the input signal.

(a) Two-Dimensional Input Signal *lena*.

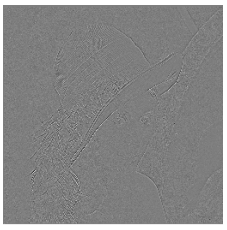
(b) Approximations.



(c) Horizontal Details.



(d) Vertical Details.



(e) Diagonal Details.

Figure 3.19: Two-Dimensional FWT Analysis.

Similarly to the one-dimensional case, a multiresolution representation can also be achieved by iterating the analysis procedure, whilst saving all the detail coefficients that result at every resolution step, and employing the resulting approximations coefficient, at the current resolution, as the input for a new resolution decomposition. The filter bank for a two-step decomposition is shown in Figure 3.20.

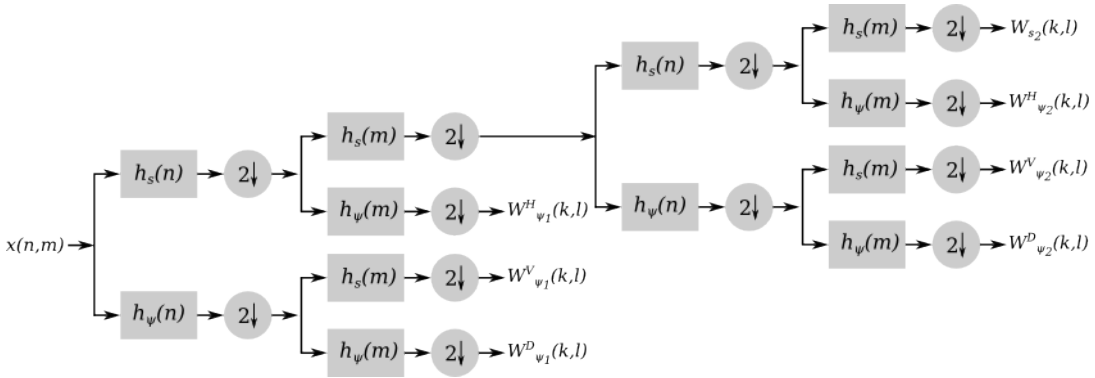


Figure 3.20: Two-Dimensional FWT Synthesis Filter Bank for a Two-Level Resolution Decomposition.

Figure 3.21 shows the size of the resulting coefficient images for a three-step decomposition.

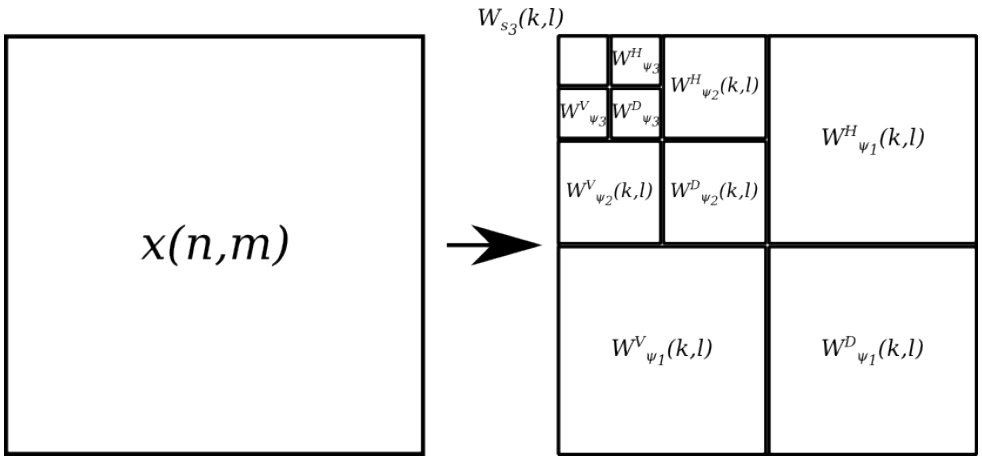


Figure 3.21: Three-Level Decomposition Coefficient Images.

Similarly to the one-dimensional scheme, the two-dimensional FWT is as well invertible. Synthesis is simply performed by the filter bank arranged in a backwards form, where the coefficients are now the inputs to a filter bank built by upsamplers and reverse filters. In the same way as in the one-dimensional FWT, synthesis from a multiresolution set of coefficient details and approximations, is performed iteratively. Figure 3.22 shows the filter bank for the inverse two-dimensional FWT.

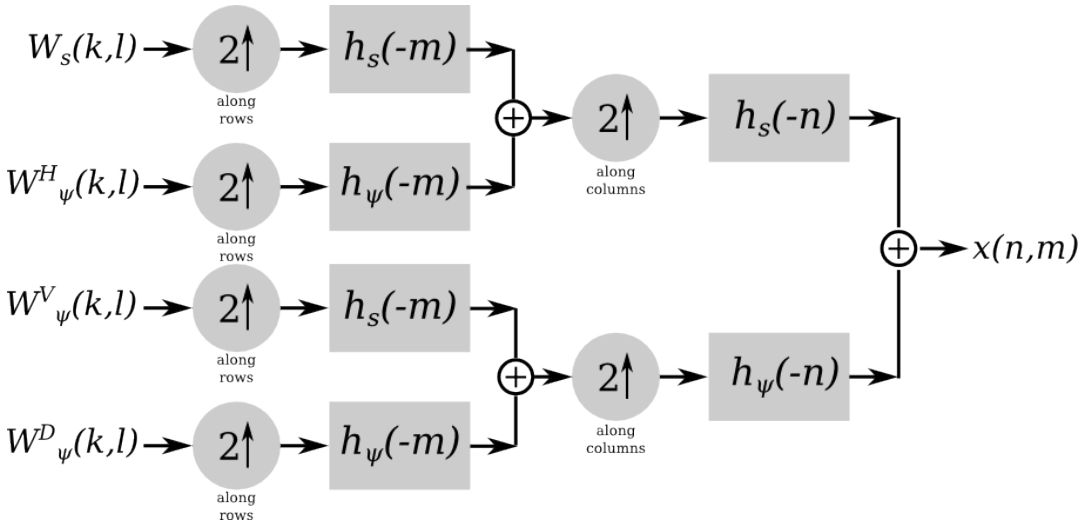


Figure 3.22: Two-Dimensional FWT Synthesis Filter Bank.

As mentioned at the beginning of this section, the two-dimensional implementation of the FWT is simply achieved through the rearrangement of

the one-dimensional scheme configuration. The two-dimensional case, is therefore a concise and fast transform, which is as well invertible.

3.6 The Hilbert-Huang Transform

Although the WT as a concise transform for the analysis of non-stationary signals is somehow new, a new alternative for the analysis of non-linear and non-stationary signals is available. Such tool is known as the Hilbert-Huang Transform (HHT) and is an empirical method whose basis claim to be adaptive so that it can be employed to produce representations not only from non-stationary signals, but also non-linear. Although the HHT has been exhaustively validated, its capabilities to analyse non-linear signals arise at the expense of difficulty in establishing a concise theoretical foundation. Nowadays, many mathematical problems related to the HHT still need to be resolved. For more information about the HHT, the reader can refer to [130].

3.7 Summary

The WT analysis of signals, allows to formulate their presentation in a range of different scales, quantified by the respective wavelet output coefficients. This allows to introduce the forward and inverse WT, which will be used in Chapter 4 in conjunction with the FBP reconstruction method to allow efficient parallel block reconstruction of the approximations and details coefficients spatial distributions. Thus the imaged object is reproduced by taking inverse WT from the multiresolution image representations.
This scenario is possible by the adoption of the multiresolution theory, whereby a discrete signal is processed with a QMF, comprised by a range of high/low pass digital filters and downsamplers. The result is a decomposition of the input signal into different resolution scale components, differing in size by a factor of two between two neighbouring scales. Low and high pass filters (i.e. scaling functions and wavelet functions, respectively) are convolved with the an input signal to produce a pair of details and approximations coefficients: approximations composed by lower frequencies, and details composed by higher frequencies. In the two diensions three details (horizontal,vertical and diagonal) and one approximations are obtained from separable products between both

scaling and wavelet functions.
A representation of a certain resolution depth, can be achieved by iterating the FWT filter bank, whilst saving all the detail coefficients that result at every resolution step, and employing the resulting approximations coefficient at the current resolution as the input for a new resolution decomposition.

Chapter 4

Wavelet-Based, Multiresolution Block Reconstruction Algorithm

As previously mentioned in this thesis, the FBP is considered as the most important reconstruction method in Tomography. It can be found in a wide range of applications, and is known to deliver high resolution images at great computational efficiency [2, 12]. Although, in order to exploit all the good characteristics that the FBP can offer, it is necessary to provide enough and good quality projection data.

The minimisation of errors caused by the incompleteness of projection data, has been an object of study from as early as the first CT developments, on a constant trend to produce images with enhanced accuracy [15]. Such developments had an impact in medical imaging, where intentionally avoiding the acquisition of large sets of projection data, and restricting the image reconstruction up to a localised ROI, was introduced in order to reduce radition dose exposure [25, 26].

Along the many different approaches to achieve the reconstruction of accurate ROIs, the wavelet-based alternative is the one that motivated the development of this work [33, 34, 39]. The purpose of this work is not only to accurately perform reconstruction of a ROI, but also to achieve a fast reconstruction algorithm while executing concurrent reconstruction of several ROIs to be merged into a single full resolution image, by using the two-dimensional FWT.

4.1 Computer Implementation of the Filtered Backprojection Method

The starting point in the development of the reconstruction algorithm presented in this thesis was the computer implementation of the FBP method, based on the formulation given in [1, 131, p. 69].
By recalling the mathematical expression shown through Equation 2.31, from the FBP review in Section 2.4.2.2, it can be seen that there are two main components that characterize the method. The first one is the filtering in Fourier domain, between projection data $R_\theta(s)$ and a weighting filter $|s|$ to produce the angular filtered projection $Q_\theta(x\cos\theta + y\sin\theta)$ (i.e. ramp filtering). The second one is the translating of such filtered projections to its corresponding points $\rho = (x\cos\theta + y\sin\theta)$, along the line integrals over the Cartesian plane, which corresponds to the image $\mu(x, y)$ (i.e. backprojection).

$$\mu(x,y) = \int_0^\pi \int_{-\infty}^{\infty} R_\theta(s)|s|e^{j2\pi s(x\cos\theta + y\sin\theta)}dsd\theta, \tag{2.31}$$

$$Q_\theta(\rho) = \int_{-\infty}^{\infty} R_\theta(s)|s|e^{j2\pi s(x\cos\theta + y\sin\theta)}ds, \tag{2.32}$$

$$\mu(x,y) = \int_0^\pi Q_\theta(x\cos\theta + y\sin\theta)d\theta. \tag{2.33}$$

So far, in the mathematical formulation of the FBP reviewed in this thesis, the projection data has been considered to be continuous, however, in order to be able to achieve a practical implementation of the FBP method, it is required that projection data be represented in a finite sampled form.

4.1.1 Projection Data Filtering

By looking at Equation 2.32, which refers to the inverse FT of the product between the Fourier domain projection data $R(s, \theta)$, and the ramp filter $|s|$, it can be noticed, from the integration limits, that the inverse FT is theoretically performed over all spatial frequencies. However, for a practical implementation, projection data must be considered to be bandlimited. If S is a frequency above

the highest frequency component within projection data, then by the Shannon theorem, projection data can be sampled at period τ without introducing any error [1].

$$\tau = \frac{1}{2S}. \tag{4.1}$$

Now, by assuming that for large values of $|\rho|$ (i.e. points along line integrals $\rho = (x\cos\theta + y\sin\theta)$), projection data equals to zero, for some large value of N, projection data in its sampled form is:

$$r_\theta(\nu\tau), \quad \nu = \frac{-N}{2}, \cdots, 0, \cdots, \frac{N}{2} - 1. \tag{4.2}$$

This way, the filtering of projection data expression, given by Equation 2.32, can be approximated to the following Discrete Fourier Transform (DFT)-based discrete from:

$$Q_\theta(\rho) \approx \frac{2S}{N} \sum_{\nu=-N/2}^{N/2} R_\theta\left(\nu\frac{2S}{N}\right) \left|\nu\frac{2S}{N}\right| e^{j2\pi\nu(2S/N)\rho} \tag{4.3}$$

Finally, if it is only required to obtain filtered projections $Q_\theta(\rho)$ for only those points ρ, at which projection data $r(\rho,\theta)$ is sampled, Equation 4.3 turns to:

$$\begin{gathered} Q_\theta\left(\tfrac{k}{2S}\right) \approx \tfrac{2S}{N} \textstyle\sum_{\nu=-N/2}^{N/2} R_\theta\left(\nu\tfrac{2S}{N}\right) \left|\nu\tfrac{2S}{N}\right| e^{j2\pi(\nu k/N)} \\ k = -N/2, \cdots, -1, 0, 1, \cdots, N/2 \end{gathered} \tag{4.4}$$

Equation 4.4 seems suitable to be computationally implemented to calculate filtered projection data, however according to [1], such expression is only valid when projection data is of finite order and bandwidth. Because those characteristics are never strictly satisfied altogether, practical implementation based on Equation 4.4 is known to be affected by interperiod interference artifacts [1]. A detailed explanation about such artifacts, as well as graphic evidence of them, can be consulted in [1, p. 69].

In order to achieve a more accurate computation of the projection data filtering component, Avinash C. Kak proposed a different approach in [131]. This alternative implementation consists in designing a bandlimited ramp filter $H(s)$, whose impulse response is constructed with the same sample interval τ as projection data $r_\theta(\nu\tau)$. The bandlimiting of ramp filter $|s|$ to above the highest

frequency component S, to obtain $H(s)$, is done through:

$$H(s) = |s| b_S(s), \tag{4.5}$$

where

$$b_S(s) = \begin{cases} 1 & |s| < S \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The FBP is then suggested to be implemented by using bandlimited ramp filter in Fourier domain (i.e. transfer function) $H(s)$. The ideal response of $H(s)$ is illustrated in the figure below:

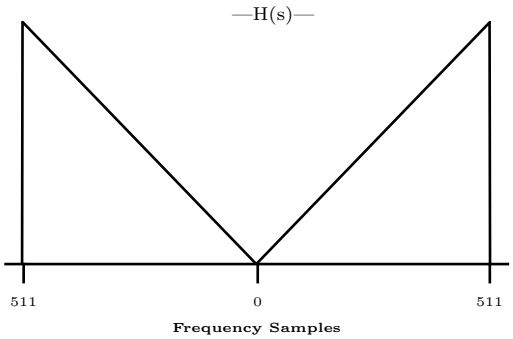


Figure 4.1: Bandlimited Ramp Filter $H(s)$.

In order to proceed with the construction of a ramp filter with the same sampling interval τ, as the projection data to be filtered, the impulse response $h(\rho)$, from the bandlimited filter's trasnfer function $H(s)$, must be obtained. Consequently, by the inverse FT of $H(s)$:

$$h(\rho) = \int_{-\infty}^{\infty} H(s) e^{j2\pi s\rho} ds, \tag{4.7}$$

which by means of sampling period τ of Equation 4.1, results in:

$$h(\rho) = \frac{1}{2\tau^2} \frac{\sin 2\pi\rho/2\tau}{2\pi\rho/2\tau} - \frac{1}{4\tau^2} \left(\frac{\sin \pi s/2\tau}{\pi s/2\tau} \right)^2. \tag{4.8}$$

From Equation 4.8, the sampled impulse response $h(n\tau)$ of $h(\rho)$ is therefore

given by:

$$h(n\tau) = \begin{cases} 1/4\tau^2, & n = 0 \\ 0, & n \text{ even} \\ -\frac{1}{n^2\pi^2\tau^2}, & n \text{ odd.} \end{cases} \tag{4.9}$$

The resulting function $h(n\tau)$ for $-5 \leq n \geq 5$ samples, constructed from Equation 4.9, is shown below.

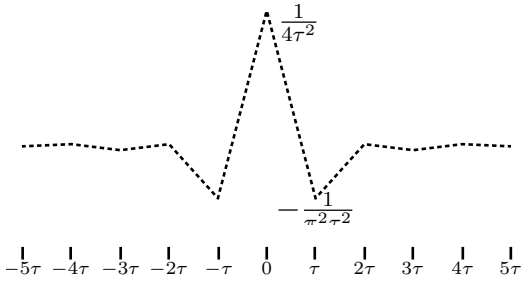


Figure 4.2: Sampled Ramp Filter Impulse Response $h(n\tau)$.

By having both projection data and the ramp filter in sampled form, as well as bandlimited, a suitable expression for practical implementation of the projection data filtering $Q_\theta(\rho)$ is therefore achievable. Equation 2.32 turns to the finite sampled $Q_\theta(n\tau)$ by:

$$Q_\theta(n\tau) = \tau \sum_{k=-\infty}^{\infty} h(n\tau - k\tau) * r_\theta(k\tau). \tag{4.10}$$

The discrete convolution of Equation 4.10 is suitable to be directly implemented in a computer, nonetheless it may be faster to be implemented in the Fourier domain through the FFT. Finally, as suggested in [1, p.74], in order to overcome the interperiod interference artifacts that are inherent to periodic convolution, projection data should be zero-padded to its nearest power of two. The Fourier domain implementation of projection data filtering is given by the following expression, where ZP refers to zero-padding:

$$Q_\theta(n\tau) = \tau \cdot \text{inverse FFT}\left\{[\text{FFT } r_\theta(n\tau) \text{ with ZP}] \cdot [\text{FFT } h(n\tau) \text{ with ZP}]\right\} \tag{4.11}$$

Kak's approach efficiency has been proved to be reliable while being the

foundation in the MATLAB's implementation of the inverse RT function: *iradon* [132]. Such function has been therefore very important in the development of the reconstruction algorithm presented in this work. The ramp filtering design component within the *iradon* function code, has been partially reproduced as shown bellow:

```
%=========================================================================
%---RAMP FILTER DESIGN---------------------------------------------------
%=========================================================================
% bandlimited ramp filter (Eqn. 61 Chapter 3, Kak and Slaney)
len = length(sngrm);
order = max(64,2^nextpow2(len)); %up to the next highest ...
    power of 2
n = 0:order; % 'order' is always even.
filtImpResp = zeros(1,order+1); %the bandlimited ramp's ...
    impulse response vector
filtImpResp(1) = 1/4; % Set the DC term
filtImpResp(2:2:end) = -1./((pi*n(2:2:end)).^2); % Set the ...
    values for odd n  (values for even n are 0)
filtImpResp = [filtImpResp filtImpResp(end-1:-1:2)];
%=========================================================================
%---RAMP FILTER IN THE FOURIER DOMAIN------------------------------------
%=========================================================================
filt = 2*real(fft(filtImpResp));
filt = filt(1:order+1);
filt = [filt.' ; filt(end-1:-1:2).'];    % Symmetry of the filter
```

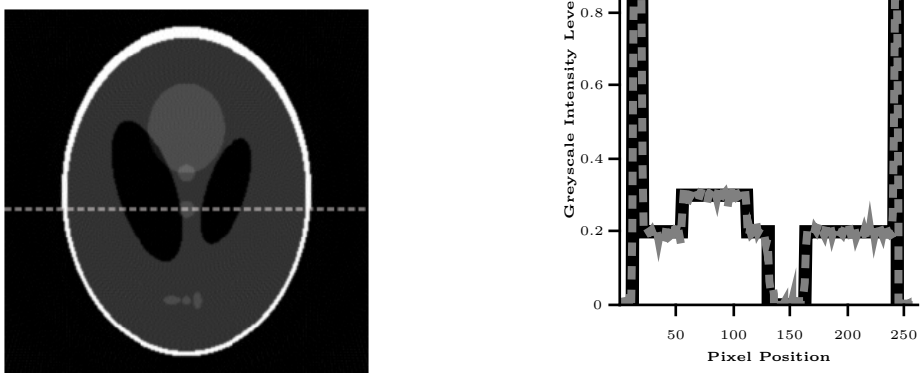
Where *len* is the length of angular projections, which is also the number of rays that form the angular projection.
The code that computes the filtering of projection data in Fourier domain, as well as the inverse FFT of the result, is shown bellow.

```
%=========================================================================
%---FT OF PROJECTION DATA------------------------------------------------
%=========================================================================
sngrm(length(filt),1) = 0;  % Zero pad projections
ft_sngrm = fft(sngrm);    % fft of projections
%=========================================================================
%---FILTERING IN FOURIER DOMAIN------------------------------------------
%=========================================================================
```

```
for i = 1:length(angles)
    filtrd_sngrm(:,i) = ft_sngrm(:,i).*filt;
end
%=========================================================================
%---INVERSE FFT OF FILTERED PROJECTION DATA----------------------
%=========================================================================
proj = real(ifft(filtrd_sngrm));     % ifft of filtered ...
    projections
proj(len+1:end,:) = [];   % Truncate the filtered projections ...
    to initial projection data size
```

where the variable *proj* is the output that corresponds to the set of filtered projection data in Radon domain, that is ready to be backprojected over the Cartesian plane to form the image. This procedure is explained in Subsection 4.1.2.

To show the accuracy of Kak's alternative method, a reconstruction of a 256 by 256 pixels Shepp-Logan phantom [133], from 180 equispaced angular projections, is shown in the Figure 4.3.



(a) Reconstructed Image.

(b) Greyscale Intesity Comparison Along Pixel Row 115.

Figure 4.3: FBP Image Reconstruction.

where Figure 4.3b is a plot corresponding to the greyscale intensity denoted by the dotted grey line in Figure 4.3a. The grey dotted line in the plot corresponds to the levels of the reconstructed image, whilst the black line in the background corresponds to the levels of the input phantom. As can be seen from the reconstruction example, Kak's approach is able to preserve the same intensity

levels of the input phantom.
In order to measure the quality of the reconstructed image, five different metrics have been adopted:

- Average Error (AVERR). This metric consists simply in the difference between the greyscale intensity value average of both the phantom and the reconstructed image.

$$AVERR = \frac{\sum_{m,n} |phantom(m,n) - image(m,n)|}{m * n}. \tag{4.12}$$

- Normalised Absolute Error (NABS). This second metric is a generalization of the AVERR, and is used to emphasize the effect of many small errors. Moreover, this metric is invariant with scaling of the phantom. According to [134], NABS is one of the most common metrics employed to measure the perceived image quality. NABS is calculated as follows.

$$NABS = \frac{\sum_{m,n} |phantom(m,n) - image(m,n)|}{\sum_{m,n} |phantom(m,n)|}. \tag{4.13}$$

- Mean Square Error (MSE). This metric is as well very similar to the AVERR, although in order to measure the amount by which the reconstructed image differ from the input phantom, the MSE calculates the difference between the average of the squares of both images. The MSE is calculated in the following manner.

$$MSE = \frac{\sum_{m,n} |phantom(m,n) - image(m,n)|^2}{m * n}. \tag{4.14}$$

- Peak Signal to Noise Ratio (PSNR). This metric, as well as the MSE, is commonly used to measure the quality between an original input image and its compressed version. The PSNR employs the MSE to derive a measure of the peak error. The lower the value of the MSE, the higher the PSNR, and hence the better quality of the reconstructed image [135]. The PSNR is calculated through the following equation.

$$PSNR = 10 \log_{10} \left(\frac{peak\_value^2}{MSE} \right). \tag{4.15}$$

- Structural Similarity Index (SSIM). The calculation of the SSIM is the most complex among the metrics employed in this thesis. It was designed as a more consistent metric with the human eye perception, compared with more common metrics like the MSE and PSNR [136]. The SSIM is a full reference metric, where the reconstructed image is compared with a reference original image, being a unity the maximum correspondence between both images. According to the MATLAB function [137], the mathematical expression for the SSIM is as follows:

$$SSIM = \frac{(2\mu_p\mu_i + C_1)(2\sigma p - i + C_2)}{(\mu_p^2 + \mu_i^2 + C_1)(\sigma_p^2 + \sigma_i^2 + C_2)}, \tag{4.16}$$

  where μ_p and μ_i are the local mean for both the phantom and the reconstructed image, σ_p and σ_i correspond to the standard deviation, and σ_{p-i} to the cross-covariance for the same images. C_1 and C_2 are regularization constants.

The following table show the quality analysis of reconstructed images with both types of ramp filters: the simple discrete approximation and the one calculated through Kak's proposed method. For both type of reconstructed images, the Shepp-Logan input phantom is used as reference image. Measurements are calculated according to the previously mentioned metrics.

Type of Ramp Filter	AVERR	NABS	MSE	PSN	SSIM
Approximated	1.005	8.1881	2.1303	-3.2845	0.0297
Kak's Method	0.0174	0.1419	0.0018	27.3788	0.8663

Table 4.1: Reconstruction Quality Comparison Between the Approximated Ramp Filter and the Kak's Method Ramp Filter.

From the results shown in Table 4.5 it is possible to see that the quality of the image obtained through Kak's method is able to deliver a considerably more accurate approximation to the input phantom image, than a simple discrete approximation of the FBP. The computer implementation proposed by Kak has been adopted in the development of our proposed custom algorithm.

4.1.2 Backprojection

Backprojection is the second component in the FBP, which consists on the mapping of all the intensity values obtained from the projection data filtering to its corresponding grid position over the $\mu(x, y)$ plane. Backprojection was previously defined in continuous domain through Equation 2.33 where filtered projections in Radon domain $Q(\rho, \theta)$ are translated to its corresponding points $\rho = (x \cos\theta + y \sin\theta)$ along the integral lines, at different angles within the range from 0° to 180°. In discrete form, Equation 2.33 can be expressed by:

$$\mu(x, y) = \frac{\pi}{K} \sum_{i=1}^{K} Q_{\theta_i}(x \cos\theta_i + y \sin\theta_i), \tag{4.17}$$

where K refers to the number of angles θ_i, for which projections $r(\rho, \theta)$ are known.
In practice, more specifically in MATLAB, discrete filtered projection data $Q(\rho, \theta)$ is usually contained within a θ-by-ρ array, which grey scale values have to be reallocated into the $\mu(x, y)$ plane.

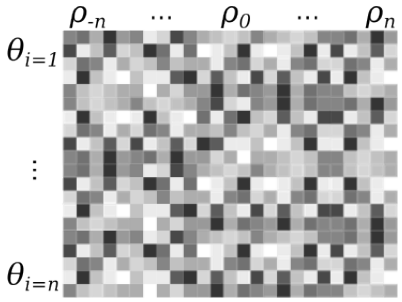


Figure 4.4: Projection Data Array $Q_\theta(t)$.

Thus if every projection angle θ_i is seen as a vector, containing n ρ elements, the task then turns to find the t location of every ρ within the θ_i vector, for the given x and y. This is done through:

$$t = (x \cos\theta_i + y \sin\theta_i), \tag{4.18}$$

where MATLAB capabilities to perform matrix operations can be exploited by building matrices $\mathbf{X}$ and $\mathbf{Y}$ containing all required x and y values, and calculate

an array $\mathbf{T}$ containing all the corresponding t indexes.

$$\mathbf{T} = \mathbf{X} \cdot \cos\theta + \mathbf{Y} \cdot \sin\theta. \tag{4.19}$$

The resulting array $\mathbf{T}$ is of the same size as $\mu(x, y)$, and is filled with the t indexes for the given angle θ_i, thus $\mathbf{T}$ contains all the necessary information for the mapping of ρ values into $\mu(x, y)$.

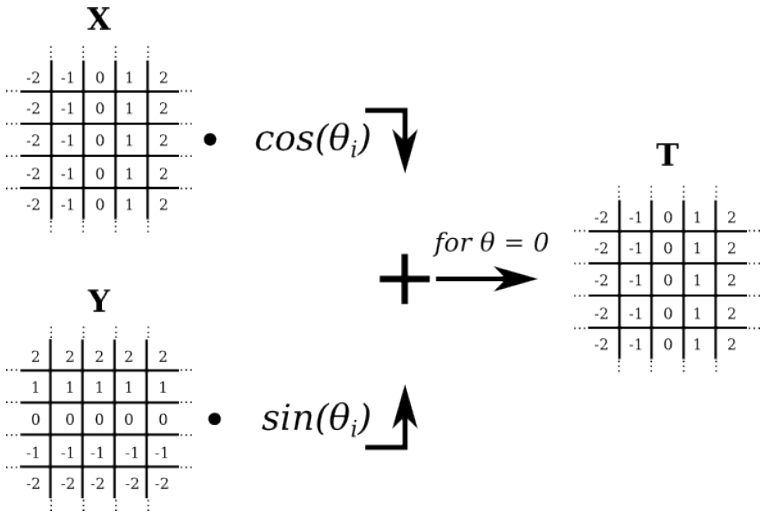


Figure 4.5: $\mathbf{T}$ index matrix calculation.

After calculating the $\mathbf{T}$, $\mu(x, y)$ is filled with the ρ values indexed by t, at the (x, y) location as from where t resides inside $\mathbf{T}$. The procedure is repeated for every θ_i and accumulated in $\mu(x, y)$ as shown in Figure 4.6.

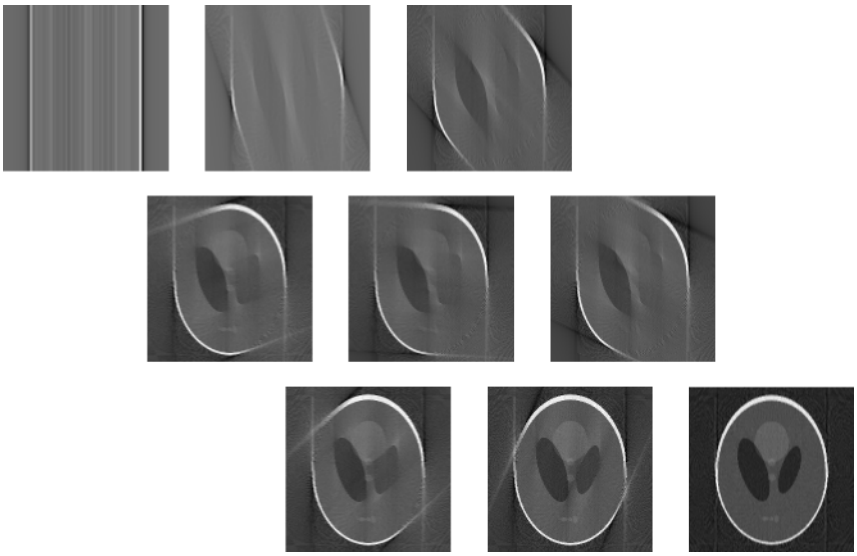


Figure 4.6: Backprojection.

MATLAB code for the backprojection computer implementation is shown below, again partially reproduced from the *iradon* function [132].
The first section of MATLAB code, corresponds to the $\mathbf{X}$ and $\mathbf{Y}$ matrices creation as well as to define the size of the output image.

```
%=================================================================
%---X AND Y CARTESIAN COORDINATES ARRAYS AND MEMORY ALLOCATION---
%=================================================================
N = size(out_image, 1);
xleft = -center + 1;
X = (1:N) - 1 + xleft;  %vector containing -center:center in ...
    the x axis
X = repmat(X, N, 1);    %matrix containing the vectors ...
    mentioned above
ytop = center - 1;
Y = (N:-1:1).' - N + ytop;  %vector containing center:-center ...
    in the y axis
Y = repmat(Y, 1, N);        %matrix containing the vectors ...
    mentioned above
ctrIdx = ceil(len/2);     % index of the center of the ...
    projections
theta = pi*angles/180;  %angles in radians
% Generate trignometric tables
```

```
costheta = cos(theta);
sintheta = sin(theta);
% Allocate memory for the image
img = zeros(N,class(p));
```

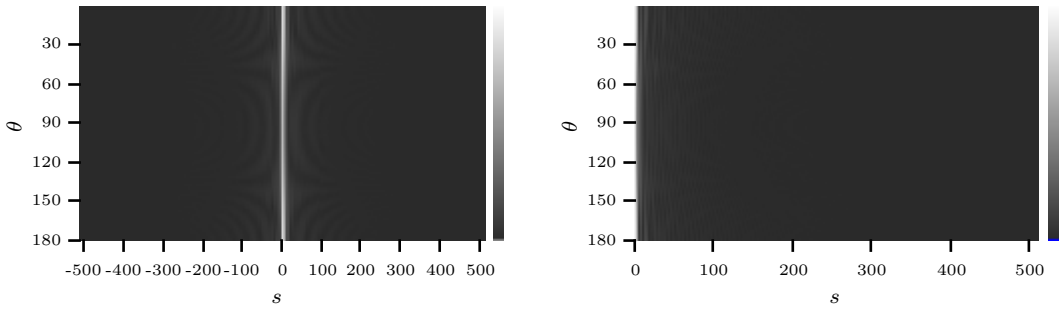
The second section of code shown bellow, performs the calculation of the **T** array as well as the mapping of all ρ values into the output image. This section of code ends by accumulating the mapped image *img* at every angle θ_i within the *for* loop.

```
1  %=================================================================
2  %---BACKPROJECTION-----------------------------------------------
3  %=================================================================
4  for i=1:length(angles)  %for every angle
5      t_proj = proj(:,i);
6      T = X*costheta(i) + X*sintheta(i);
7      A = floor(T);   %round t indexes to get integer indexes
8
9      img = img + (t-a).*t_proj(a+1+ctrIdx) + ...
           (a+1-t).*t_proj(a+ctrIdx); %fill the (x,y) plane with ...
           linear interpolation and accumulate with previous angle
10     image = img*(pi/(2*length(theta))); %Normalisation of ...
           intensity leves (K value in Discrete formula given by ...
           Kak and Slaney)
11 end
```

As can be seen from the code above, the calculation of **T** will never result into integer values, therefore results are rounded toward negative infinity, through the *floor* function. The error added in the rounding of t values, is alleviated with the linear interpolation as suggested by [1, p.67], done while mapping ρ values into *img* (i.e. line 9 in the code above).
Although the computer implementation of the backprojection is not as complex as the filtering of projection data, it is the most computationally expensive component within the FBP method, so its simplification has been object of study in the pursuit to achieve image reconstructions at higher speed rates [138].

4.2 First Approach to Computation Complexity Reduction of the FBP

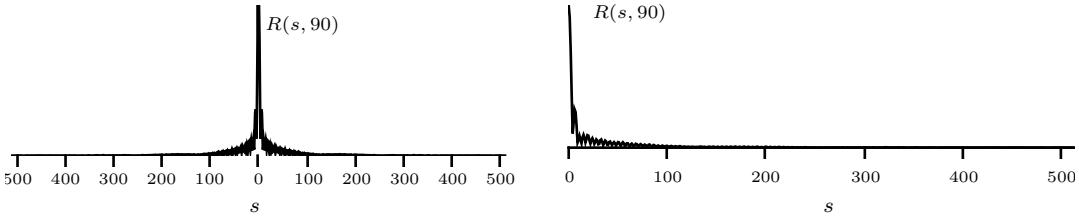
After having achieved a successful implementation of the FBP method, the first approach to reduce the computation complexity of the method, and hence a faster reconstruction, was achieved by simply exploiting the symmetry of FFT calculations in the filtering component. Fourier domain projection data $R(s, \theta)$, as well as the ramp filter $H(s)$ (i.e. $|s|$ from the FBP definition expression), involve FFT calculations in its computer implementations, therefore truncation of such Fourier domain data, to the positive frequency spectrum, can be exploited to reduce the amount operations involved in the filtering component, without harming the quality of the output image.
An example in which projection data $R(s, \theta)$ and the ramp filter $H(s)$ are truncated for the computation of filtered projection data $Q(\rho, \theta)$, is presented below.
Figure 4.7 shows the absolute values of Fourier domain projection data $R(s, \theta)$, for both the full and only-positive frequency spectrum. Such data was obtained through the FFT of a 180-angles RT, of a $256x256$-pixel Shepp Logan phantom.



(a) Full Spectrum ($-512 \leq s \geq 512$) (b) Positive Spectrum ($0 \leq s \geq 512$).

Figure 4.7: Absolute Values of the Fourier Domain Projection Data $R(s, \theta)$.

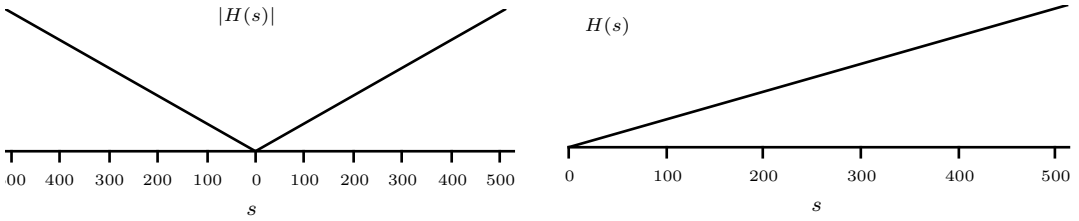
In order to obtain a different perspective of the data truncation, the Figure 4.8 shows the same absolute values, but only for a 90° slice, taken from the previously shown projection data set $R(s, \theta)$.



(a) Full Spectrum ($-512 \leq s \geq 512$) (b) Positive Spectrum ($0 \leq s \geq 512$).

Figure 4.8: Absolute Values of a Fourier Domain Projection Data Slice at 90°, $R_{\theta_{90}}(s)$.

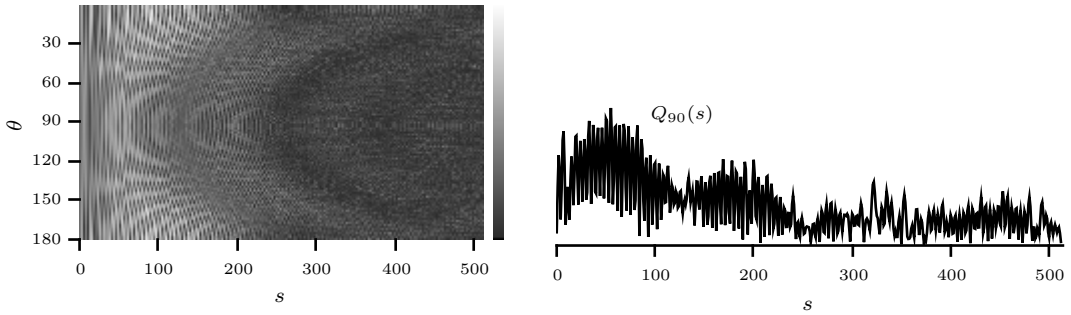
The same operation is therefore applied to the ramp filter $H(s)$, to obtain the function shown in Figure 4.10b, shown below:



(a) Full Components ($-512 \leq s \geq 512$) (b) Positive Components ($0 \leq s \geq 512$).

Figure 4.9: Absolute Values of the Fourier Domain Ramp Filter $H(s)$.

In order to produce the filtered projection data in Fourier domain $Q(s, \theta)$, a product between Fourier domain projections $R(s, \theta)$ and ramp filter $H(s)$ is accomplished by only employing data in the range $0 \leq s \leq 512$. Absolute value plots for the resulting $Q(s, \theta)$, as well as for the 90° slice, are shown below:

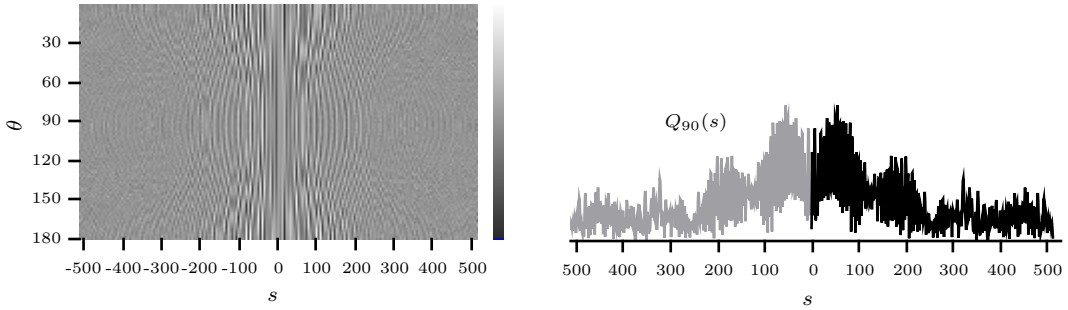


(a) Filtered Projection Data $Q(s, \theta)$. (b) Filtered Angular Projection $Q_{\theta_{90}}(s)$.

Figure 4.10: Filtered Projection Data from Truncated $R(s, \theta)$ and $H(s)$.

After filtering projection data, by working only with samples within the positive

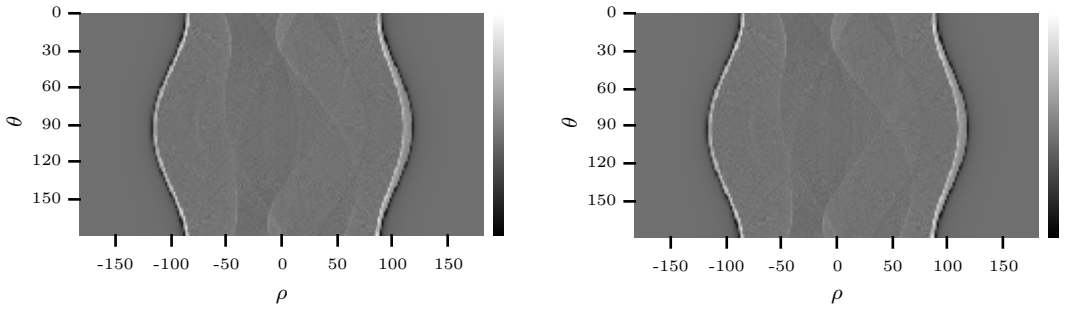
frequency spectrum, the negative frequency components must be added in order to proceed with the inverse FFT calculation to obtain filtered projection data $Q(\rho, \theta)$. The negative frequency spectrum data completion is simple achieved by adding a reverse order and complex conjugated version of positive frequency spectrum data. Such operation is shown below, again for the complete projection data set $R(s, \theta)$, and for the 90° angular projection $R_{\theta_{90}}(s)$.



(a) Filtered Projection Data $Q(s, \theta)$. (b) Filtered Angular Projection $Q_{\theta_{90}}(s)$.

Figure 4.11: Complete Fourier Domain Filtered Projection Data.

At this point, the inverse FFT of filtered projection data $Q(s, \theta)$ can be carried on, so the Radon domain projection set $Q(\rho, \theta)$ (i.e. filtered sinogram) is finally obtained to perform backprojection. The following figure shows the filtered sinograms obtained in both cases, from the processing of half frequency spectrum data and from full frequency spectrum.



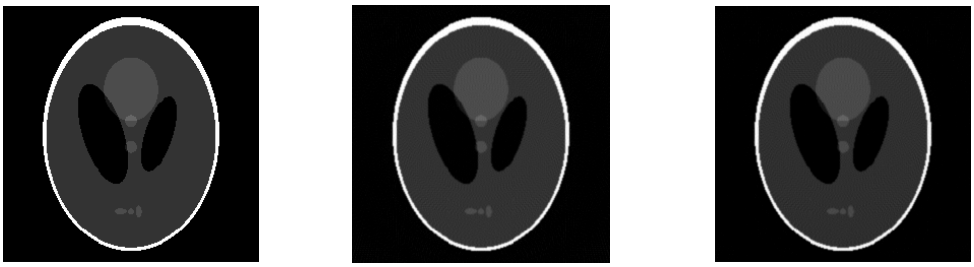
(a) Filtered Sinogram from Truncated $R(s, \theta)$ and $H(s)$.

(b) Filtered Sinogram from Complete $R(s, \theta)$ and $H(s)$.

Figure 4.12: Filtered Projection Data $Q(\rho, \theta)$.

The obtained filtered sinograms seem to be identical, as no differences are noticeable between them. Although, to have a better description of the

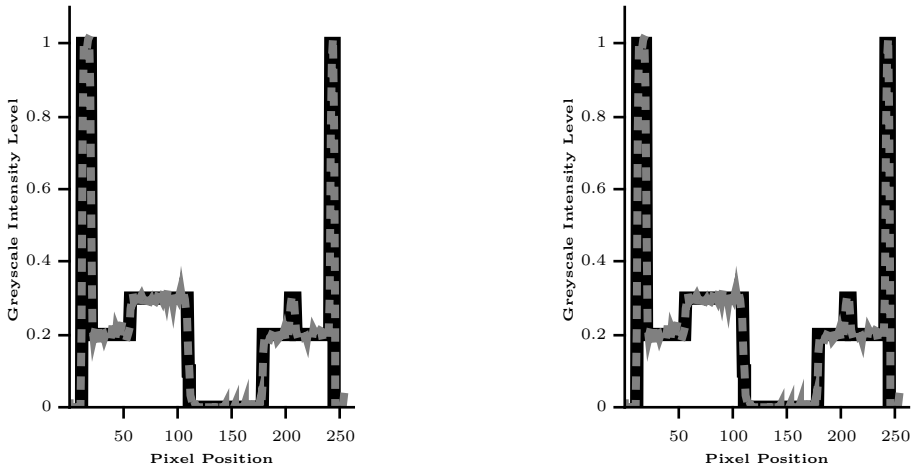
correspondence between both approaches, images are reconstructed from both sinograms. The following figure shows the input Shepp Logan phantom and both reconstructed images.



(a) Input Phantom. (b) Image from Full Spectrum Filtering. (c) Image from Half Spectrum Filtering.

Figure 4.13: Input Phantom and Reconstructed Images Comparison.

Again, as in the case of the filtered sinograms, reconstructed images show no difference either in intensity levels and structure. To have a more certain sense about the possible differences between both approaches, the intensity levels of corresponding to the row $y = 155$ is plotted for both reconstructed images, compared with the input phantom. In both plots, the black line corresponds to the phantom pixel row, while the grey dotted lines correspond to the reconstructed images.



(a) Pixel Row Plot from Full Spectrum. (b) Pixel Row Plot from Half Spectrum.

Figure 4.14: Greyscale Intensity Levels along Row $y = 155$.

It can be noticed from the comparison between plots, presented in Figure 4.14, that the reconstructed images obtained from both half and full frequency

spectrum filtering, behave in identical manner.
The following table show numerical results about the quality of both reconstructions by taking the input phantom as reference image.

Considered Frequency Spectrum	AVERR	NABS	MSE	PSN	SSIM
Full	0.0174	0.1419	0.0018	27.3788	0.8663
Half	0.0174	0.1419	0.0018	27.3788	0.8663

Table 4.2: Reconstruction Quality Comparison of the Full and Half Frequency Spectrum Image Reconstructions.

As it was expected, the error of both approaches against the input phantom is identical. The next table show the same numerical comparison between both reconstructed images.

AVERR	NABS	MSE	PSN	SSIM
0	0	0	315.9154	0.8663

Table 4.3: Error Metrics Between Both Images Reconstructed by Considering the Full and Half Frequency Spectrum.

As no error was found that may harm the reconstructed image, the half frequency spectrum approach has been considered in the development of the algorithm presented in this thesis. Although the complexity reduction here explained, does not seem to represent a big saving in the computer implementation of the FBP, the opposite results for the hardware implementation, where the reduction in the utilisation of the complex multipliers that employed in the filtering component, is traduced into the less storage and reduced logic resources, as presented in Chapter 5.
The customisation of the MATLAB *iradon* function, to consider only the positive frequency spectrum in the filtering stage of the FBP is as follows, where the ramp filter in Fourier domain is truncated.

```
1 %=======================================================================
2 %---RAMP FILTER IN THE FREQUENCY DOMAIN-------------------------
3 %=======================================================================
4 filt = (fft(filtImpResp));
5 filt(order+2:end) = [];  %Truncation of the Filter Frequency ...
      Spectrum
```

The same truncation is applied to the projection data in Fourier domain.

```
%=========================================================================
%---FT OF PROJECTION DATA-----------------------------------------
%=========================================================================
sngrm(size(filtImpResp,2),:) = 0;  % Zero pad projections
sngrm = sngrm';
for i = 1:size(sngrm,1)
    ft_sngrm(i,:) = fft(sngrm(i,:));    % fft of projections
end
ft_sngrm = ft_sngrm(:,1:order+1); %Truncation of Projection ...
   Data Frequency Spectrum
```

The filtering in Fourier domain is therefore performed through the point-to-point product between the half frequency spectrum ramp filter and projection data. After filtering, the full frequency spectrum is completed by adding a complex conjugated reverse order version of filtering data just obtained.

```
%=========================================================================
%---FILTERING IN FOURIER DOMAIN-----------------------------------
%=========================================================================
for i = 1:length(angles)
    f_projections(i,:) = filt.*ft_sngrm(i,:); % frequency ...
       domain filtering
end
f_projections = [f_projections ...
   conj(f_projections(:,end-1:-1:2))];
```

Finally, the inverse FFT of the full frequency spectrum filtered data is obtained, so filtered data in Radon domain is ready to be backprojected.

```
%=========================================================================
%---INVERSE FFT OF FILTERED PROJECTION DATA----------------------
%=========================================================================
for i = 1:size(ft_sngrm,1)
    s_projections(i,:) = real(ifft(f_projections(i,:))); ...
       %ifft of filtered projections
end
```

```
7 s_projections(:,(len)+1:end) = []; % Truncate the filtered ...
      projections to its initial size (size of radon transformed ...
      data)
8 s_projections = s_projections';
```

The reconstructed images shown in Figure 4.13, as well as the plots in Figure 4.14, were obtained by employing the custom MATLAB code just explained. The full M-file is included in the Appendix A.2.

4.3 Wavelet-Based FBP

Since its appearance, the WT has received much interest by a wide range of engineering as well as research communities. In Tomography, the WT has been employed as an alternative tool to alleviate the problems associated with the FBP reconstruction from incomplete projection data [139].
As previously discussed in Chapter 2, the sources that drive projection data to incompleteness, can either be unintended or consciously induced. The first case is when data incompleteness depend upon the physical set-up and/or other restrictions in the acquisition environment. In the second case, data incompleteness is caused with the purpose of limiting the amount of acquired data, to a specific region. Such action is expected to result in complexity reduction, and hence make it possible to achieve faster image reconstructions [21, 23, 28–34].
Unfortunately, the FBP reconstruction method is known to be globally dependent upon the complete set of projection data [2, 139], therefore the limitation of angular projections (*i*.e. projection data truncation) to consider only those line integrals passing over a ROI will reflect severe harming effects in the reconstructed image. In the literature, such globally dependency of the FBP as the inversion method of the RT is commonly referred to as *non-locality* [21, 23, 140].
The non-locality of the FBP method is attributed to the filtering component, in which a ramp filter that possess a discontinuity at the origin is employed. Such a filter causes projection data to spread its support, thus after being filtered, projection data is no longer compactly supported (i.e. locally) [139].
According to Tim Olson and Joe DeStefano, who in [27] developed the first Tomography numerical algorithm that involved wavelet functions, in order to

avoid the support spread of projection data after being filtered, it is necessary to employ basis functions whose support is essentially compact and does not spread after filtering. Wavelets are functions that possess a small support and can be constructed with as many zero moments as required, so its support is essentially unchanged after being filtered [27]. For deeper details, as well as a mathematical prove of such statement, please refer to [23, 27, 139].

In [27], Olson and DeStefano employed the one-dimensional WT to analyse projection data and localise the RT, achieving to reduce the radiation exposure whilst accurately reconstructing only a ROI. Just a few years later in [33], Alexander H. Delaney and Yoram Bresler generalised Olson and DeStefano's work with the inclusion of the two-dimensional FWT. In this extended approach, Delaney and Bresler realised that the two-dimensional WT, not only preserves the same localisation properties of Olson-DeStefano's approach, but also that it naturally conducts to a multiresolution reconstruction scheme. Both Olson-DeStefano's and Delaney-Bresler's approaches rely on a ROI reconstruction framework in which the extracted local values from the projection data set that are directly involved in the desired ROI are complemented with a few sparse projection data, global values. That is, a reduced amount of line integrals, that do not pass over the ROI, are required for the proper reconstruction of such ROI.

A very similar approach that also employs the two-dimensional FWT, was proposed in [34] by Farrokh Rashid-Farrokhi, *et al.*, based on the attributes that some wavelet basis functions possess, to preserve its compact support after being ramp filtered. In contrast to Olson-DeStefano's and Delaney-Bresler's approaches, in [34] the local measurements were not only extracted from the projection data high resolution components, but also the low resolution components are truncated. As this approach claimed to achieve higher exposure reduction than the others. The authors referred to it as *essentially local reconstruction*.

For the algorithm presented in this thesis, similarly to Delaney and Bresler's approach, the filtering component of the FBP reconstruction method is generalised with the incorporation of the two-dimensional FWT.

4.3.1 The Modified Wavelet Ramp Filter

The most noticeable feature of the wavelet-based FBP arises from the generalisation of the filtering component, which turns the standard reconstruction procedure from the obtaining of a single full resolution image, to the obtaining of a set of lower resolution coefficient images. The main contribution of such a characteristic is that the backprojection of FWT coefficient images is performed over a reduced pixel grid, which if implemented in parallel, results into an increase of the overall reconstruction speed. Another benefit of the two-dimensional FWT is that a multiresolution representation of projection data can be obtained, so by having approximations and details images belonging to different scales, the choice to reconstruct images up to a desired scale is available.
To explain how the FBP filtering component is modified to incorporate the two-dimensional FWT, the mathematical expression of the FBP in its continuous form, is once again analysed:

$$\mu(x,y) = \int_{0}^{\pi} \int_{-\infty}^{\infty} R_\theta(s)|s| e^{j2\pi s(x\cos\theta + y\sin\theta)} ds d\theta, \tag{2.31}$$

where the ramp filter $|s|$ is substituted by the product between the same ramp filter $|s|$, and the two-dimensional FWT wavelet and scaling functions $\mathcal{W}$ [33, 34]. The wavelet-based filtering turns Equation 2.31 to:

$$W_{coeff}(x,y) = \int_{0}^{\pi} \int_{-\infty}^{\infty} R_\theta(s) \left[|s| \cdot \mathcal{W}(s\cos\theta, s\sin\theta)\right] e^{j2\pi s\rho} ds d\theta. \tag{4.20}$$

Equation 4.20 is the mathematical expression for the wavelet-based FBP, where $\mathcal{W}$ refers to each of the two-dimensional FWT filters in Fourier domain and polar coordinates. According to the definition of the two-dimensional FWT reviewed in Chapter 3, filters are obtained through the separable product between wavelet and scaling functions ψ and s, along rows and columns m and

n.

$$\begin{aligned} \psi^H(n,m) &= s(n)\psi(m) \quad \text{Horizontal}, \\ \psi^V(n,m) &= \psi(n)s(m) \quad \text{Vertical}, \\ \psi^D(n,m) &= \psi(n)\psi(m) \quad \text{Diagonal}, \end{aligned} \tag{3.15}$$

$$s(n,m) = s(n)s(m) \quad \text{Approximations}. \tag{3.16}$$

Both wavelet and scaling functions involved in the separable products shown in Equations 3.15 and 3.16, and originally expressed in Cartesian coordinate system, must be translated to the Fourier domain and polar coordinates in order to match with the format of the FBP filtering component. Wavelet and scaling functions in Fourier domain and polar coordinates are expressed by:

$$\begin{aligned} h_s(n) &\rightarrow H_s(u) \rightarrow H_s(s\cos\theta) \\ h_\psi(m) &\rightarrow H_\psi(v) \rightarrow H_\psi(s\sin\theta) \\ h_s(m) &\rightarrow H_s(v) \rightarrow H_s(s\sin\theta) \\ h_\psi(n) &\rightarrow H_\psi(u) \rightarrow H_\psi(s\cos\theta) \end{aligned} \tag{4.21}$$

It is important to note that after regridding the wavelet and scaling functions to polar coordinates, the wavelet modified ramp filters become angle dependent, so in the wavelet-based FBP every angular projection corresponds to a specific modified ramp filter at the same angle. The wavelet and scaling functions separable products, to obtain two-dimensional FWT filters in Fourier domain and polar coordinates, turns from Equations 3.15 and 3.16 to:

$$\begin{aligned} \mathcal{W}_\psi^H &= H_s(s\cos\theta)H_\psi(s\sin\theta) \quad \text{Horizontal}, \\ \mathcal{W}_\psi^V &= H_\psi(s\cos\theta)H_s(s\sin\theta) \quad \text{Vertical}, \\ \mathcal{W}_\psi^D &= H_\psi(s\cos\theta)H_\psi(s\sin\theta) \quad \text{Diagonal}, \end{aligned} \tag{4.22}$$

$$\mathcal{W}_s = H_s(s\cos\theta)H_s(s\sin\theta) \quad \text{Approximations}. \tag{4.23}$$

Where $H, V,$ and D, refer to the horizontal, vertical, and diagonal directions, at which every wavelet filter measure high frequency variations. The separable products from Equations 4.22 and 4.23, can then be included into the filtering component of Equation 4.20, to formulate the wavelet-based FBP. The standard FBP image reconstruction expressed by Equation 4.20, turns then to

the reconstruction of four different coefficient images: approximations $W_s(k,l)$ and details $W_\psi^H(k,l)$, $W_\psi^V(k,l)$, and $W_\psi^D(k,l)$.

$$W_s(k,l) = \int_0^\pi \int_{-\infty}^{\infty} R_\theta(s) \left[|s| \cdot \mathcal{W}_s\right] e^{j2\pi w\rho} dw d\theta \tag{4.24}$$

$$W_\psi^H(k,l) = \int_0^\pi \int_{-\infty}^{\infty} R_\theta(s) \left[|s| \cdot \mathcal{W}_\psi^H\right] e^{j2\pi w\rho} ds d\theta \tag{4.25}$$

$$W_\psi^V(k,l) = \int_0^\pi \int_{-\infty}^{\infty} R_\theta(s) \left[|s| \cdot \mathcal{W}_\psi^V\right] e^{j2\pi w\rho} ds d\theta \tag{4.26}$$

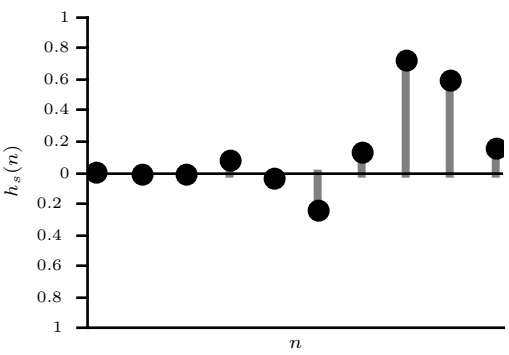
$$W_\psi^D(k,l) = \int_0^\pi \int_{-\infty}^{\infty} R_\theta(s) \left[|s| \cdot \mathcal{W}_\psi^D\right] e^{j2\pi w\rho} ds d\theta \tag{4.27}$$

In a computer implementation, wavelet and scaling functions are not necessarily required to be calculated during runtime, so if they are precomputed and stored in memory, the inclusion of the FWT into the standard FBP, does not represent an increase in speed and/or algorithm complexity. This is strictly accomplished when assuming a parallel computation of output coefficient images.
In MATLAB, the computation of the wavelet filter coefficients is similarly treated as the ramp filter previously reviewed. The first step is to choose the wavelet basis to be employed and adapt its coefficients to the format required for a FFT calculation, including zero-padding. In the following MATLAB code example, the *daubechies* wavelet with support-5 is chosen and translated to the Fourier domain through the *FFT* function.

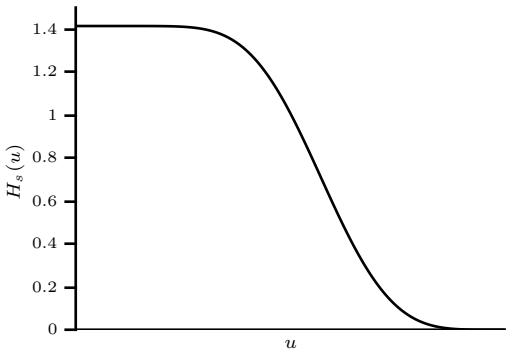
```
%==========================================================
%---WAVELET FILTERS-----------------------------------------
%==========================================================
[lp, hp] = wavefilter('db5', 'd'); %load analysis wavelet and ...
    scaling functions
%low-pass filters(scaling functions)
lp_ft1 = lp((length(lp)/2)+1:end);
lp_ft1(order*2) = 0;
lp_ft2 = lp(1:(length(lp)/2));
lp_ft1((end-(length(lp_ft2)))+1:end) = lp_ft2;
```

```
10 lp = lp_ft1; %adjusting of the filters format and zero ...
       padding for FFT
11 %high-pass filters(wavelet functions)
12 hp_ft1 = hp((length(hp)/2)+1:end);
13 hp_ft1(order*2) = 0;
14 hp_ft2 = hp(1:(length(hp)/2));
15 hp_ft1((end-(length(hp_ft2)))+1:end) = hp_ft2;
16 hp = hp_ft1;
17 %=========================================================================
18 %---FT OF WAVELET FILTERS-------------------------------------
19 %=========================================================================
20 f_lp = fftshift(fft(lp));
21 f_hp = fftshift(fft(hp));
```

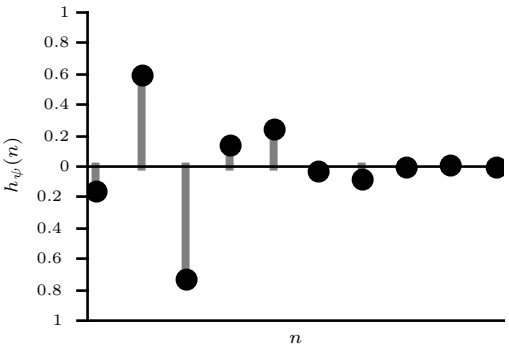
The *wavefilter* function, within the piece of code shown above, is a simple script to call the low-pas and high-pass filter coefficients that belong to a specific wavelet basis, and was taken from the FWT computer implementation covered in [129]. The support-5 *daubechies* filters, as well as their Fourier domain representation, calculated by using the MATLAB code example, are shown in Figure 4.15:



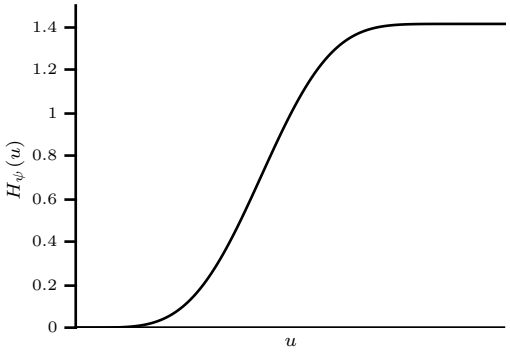
(a) *db5* Low-Pass Filter Coefficients.



(b) *db5* Low-Pass Transfer Function.



(c) *db5* High-Pass Filter Coefficients.



(d) *db5* High-Pass Transfer Function.

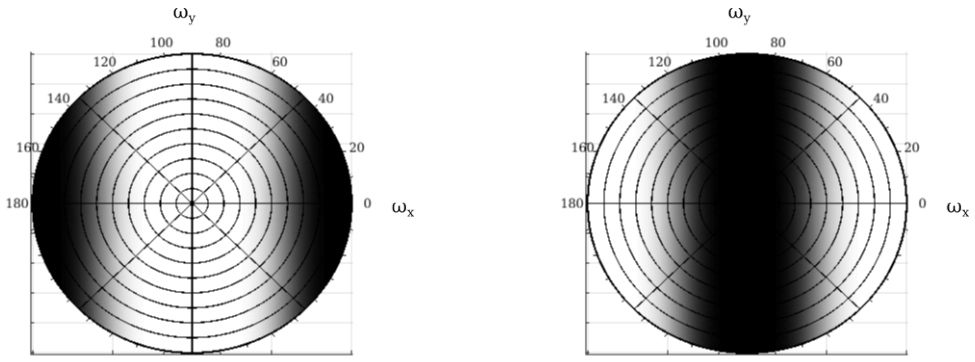
Figure 4.15: Support 5 *daubechies* Scaling and Wavelet functions.

The computation of the regridding from the Cartesian to polar coordinate systems is performed as follows:

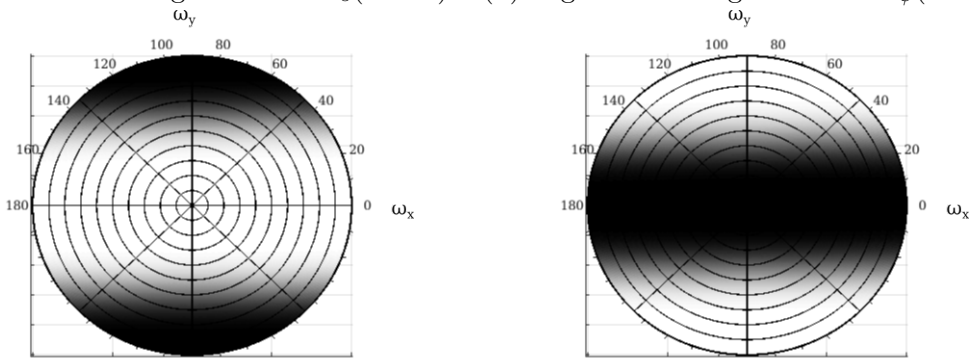
```
%=====================================================================
%---CARTESIAN TO POLAR-------------------------------------------
%=====================================================================
theta = pi*angles/180; %generate trigonometric tables
costheta = cos(theta);
sintheta = sin(theta);
s = -order:order-1;
for i=1:length(costheta)
    uu(i,:) = costheta(i).*s;
    vv(i,:) = sintheta(i).*s;
end
u = floor(uu)+order+1;
v = floor(vv)+order+1;
nl = f_lp(u);   %low-pas along rows
nh = f_hp(u);   %high-pas along rows
```

```
ml = f_lp(v);    %low-pas along columns
mh = f_hp(v);    %high-pas along columns
nh(:,1:order-1) = []; %truncate Fourier domain filters to ...
nl(:,1:order-1) = []; %consider only positive frequencies
mh(:,1:order-1) = [];
ml(:,1:order-1) = [];
mh = conj(mh);
ml = conj(ml);
```

From which the Fourier domain low-pass and high-pass filters along rows and columns, and in polar coordinates are obtained. The next figure shows the wavelet and scaling filter coefficients in the Fourier space.



(a) Low-Pass Along Columns $H_s(s\sin\theta)$. (b) High-Pass Along Columns $H_\psi(s\sin\theta)$.



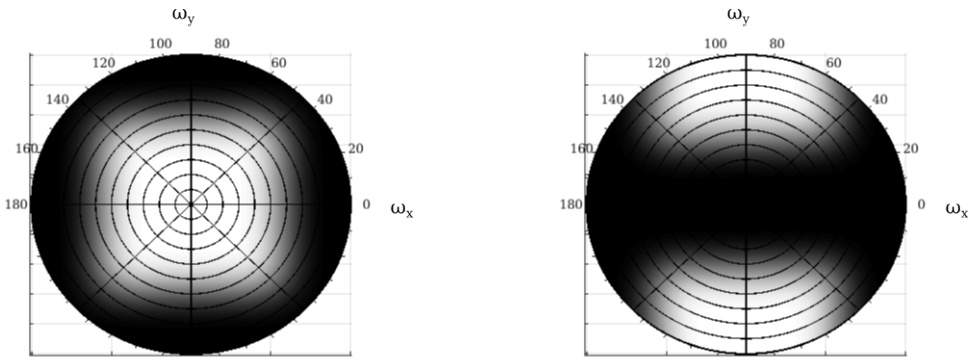
(c) Low-Pass Along Rows $H_s(s\sin\theta)$. (d) High-Pass Along Rows $H_\psi(s\sin\theta)$.

Figure 4.16: Wavelet and Scaling Functions in Polar Coordinates.

Those filters can then be directly used to create the two-dimensional FWT filters, through the separable products previously shown in Equations 4.23 and 4.22.

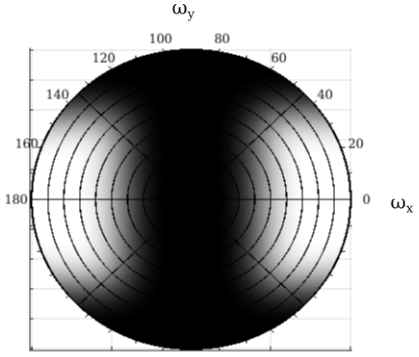
```
%=========================================================================
%---2D FWT FILTERS ------------------------------------------
%=========================================================================
app_filt = nl.*ml;  %approximations filter
hor_filt = nl.*mh;  %horizontal details filter
vert_filt = nh.*ml; %vertical details filter
diag_filt = nh.*mh; %diagonal details filter
```

From which the following four two-dimensional filters, in Fourier domain and polar coordinates are obtained:

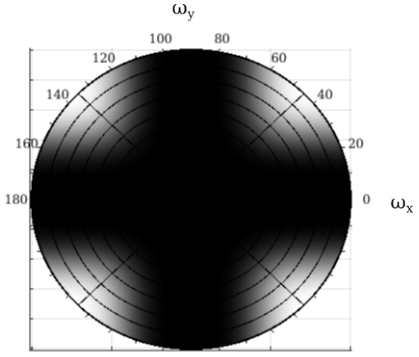


(a) Approximations.

(b) Horizontal Details.



(c) Vertical Details.



(d) Diagonal Details.

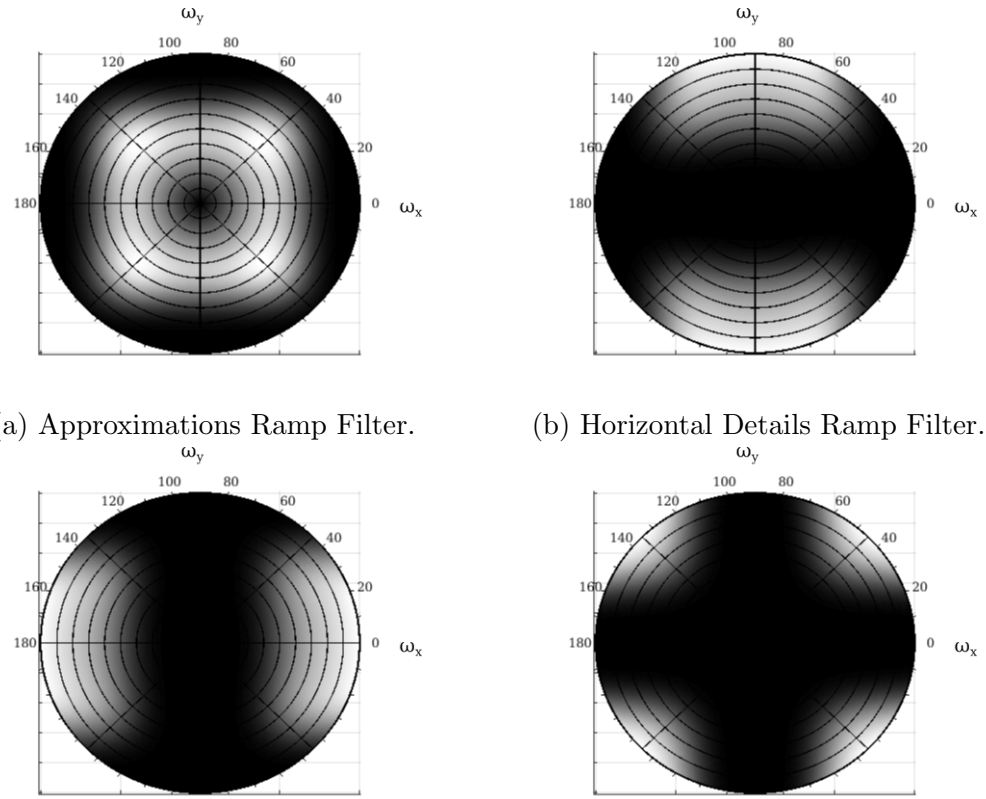
Figure 4.17: Two Dimensional FWT Filters in Polar Coordinates.

And finally multiplied with the ramp filter to create the modified ramp filters that are used to separate projection data into approximations and details coefficients.

```
%=========================================================================
```

```
%---2D FWT RAMP FILTERS ---------------------------------------
%===================================================================
app_ramp = app_filt.*filt; %approximations ramp filter
hor_filt = hor_filt.*filt; %horizontal details ramp filter
vert_filt = vert_filt.*filt; %vertical details ramp filter
diag_filt = diag_filt.*filt; %diagonal details ramp filter
```

It is possible to see, that as predicted in [27], after being ramp filtered, wavelet and scaling functions remained essentially unchanged. Modified ramp filters are shown in the next figure.



(a) Approximations Ramp Filter.

(b) Horizontal Details Ramp Filter.

(c) Vertical Details Ramp Filter.

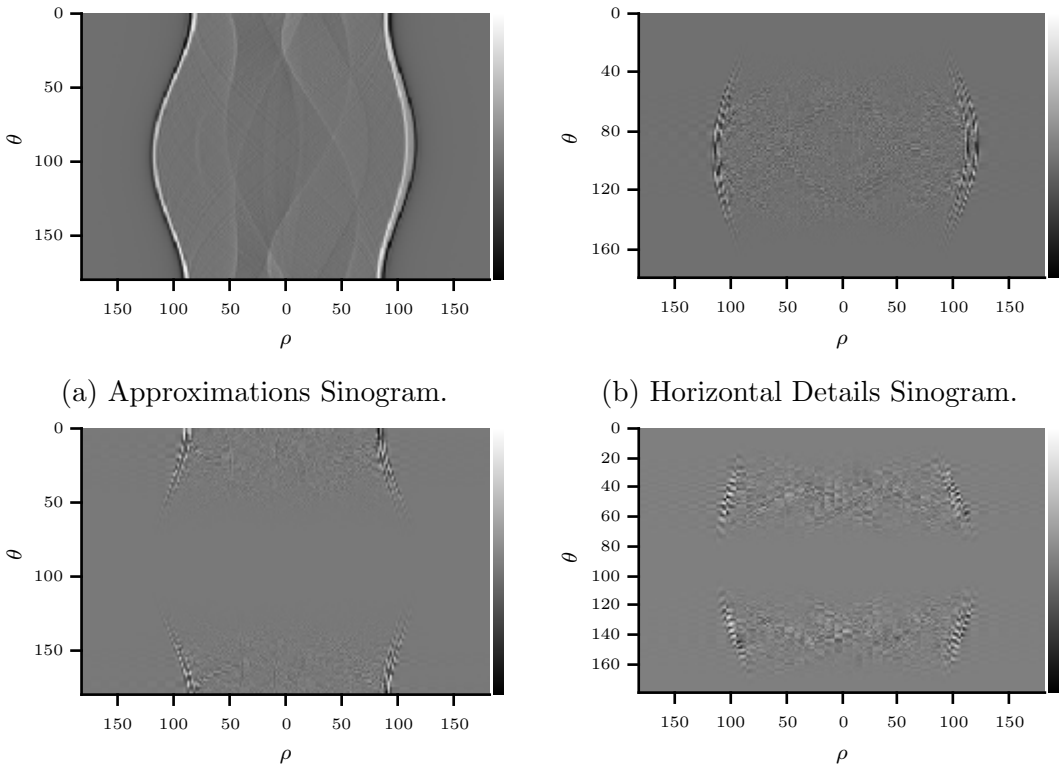
(d) Diagonal Details Ramp Filter.

Figure 4.18: Two Dimensional Modified Ramp Filters.

After calculating the modified ramp filter coefficients, the filtering of projection data is performed in the same way as was described for the standard FBP, with the difference that in this case, four different sets of filtered projection data will be obtained. The MATLAB code shown below, demonstrates how every wavelet-based sinogram is obtained from each of the different modified filters.

```
%==================================================================
%---WAVELET RAMP FILTERING IN FOURIER DOMAIN-------------------
%==================================================================
for i = 1:length(angles)
    hor_filt(i,:) = ft_sngrm(i,:).*hor_ramp(i,:);
    vert_filt(i,:) = ft_sngrm(i,:).*vert_ramp(i,:);
    diag_filt(i,:) = ft_sngrm(i,:).*diag_ramp(i,:);
    app_filt(i,:) = ft_sngrm(i,:).*app_ramp(i,:);
end
%============================negative frequencies==============
hor_filt = [hor_filt conj(hor_filt(:,end-1:-1:2))];
vert_filt = [vert_filt conj(vert_filt(:,end-1:-1:2))];
diag_filt = [diag_filt conj(diag_filt(:,end-1:-1:2))];
app_filt = [app_filt conj(app_filt(:,end-1:-1:2))];
%==================================================================
%---IFFT OF FILTERED PROJECTION DATA-------------------
%==================================================================
for i = 1:size(ft_sngrm,1)
    w_diag(i,:) = real(ifft(diag_filt(i,:)));
    w_vert(i,:) = real(ifft(vert_filt(i,:)));
    w_hor(i,:) = real(ifft(hor_filt(i,:)));
    w_app(i,:) = real(ifft(app_filt(i,:)));
end
%removal of added data for fft
w_diag(:,(len)+1:end) = []; %diagonal details sinogram
w_vert(:,(len)+1:end) = []; %vertical details sinogram
w_hor(:,(len)+1:end) = []; %horizontal details sinogram
w_app(:,(len)+1:end) = []; %approximations sinogram
```

Filtered projection data results into the set of approximations and details sinograms shown in Figure 4.19. Such figure clearly illustrates how approximations sinogram is the only one that holds data (i.e. low frequencies) from all of the angular projections. On the contrary, the angular projections involved in the details sinograms (i.e. high frequencies), are not present within the overall angular range, so the computation of a certain number of angular projections can be discarded without affecting the quality of the reconstructed image. This characteristic has been particularly taken into consideration for the development of our algorithm, and will be explained in more detail in the following subsection.



(a) Approximations Sinogram.

(b) Horizontal Details Sinogram.

(c) Vertical Details Sinogram.

(d) Diagonal Details Sinogram.

Figure 4.19: Two-Dimensional FWT Domain Sinograms.

The set of wavelet-based FBP filtered sinograms shown in Figure 4.19, are ready to be used by the backprojection component to obtain the set of FWT coefficient images.

4.3.2 Two-Dimensional FWT Coefficient Images Backprojection

In the wavelet-based FBP, the backprojection component is computed under the same principles as it is done in the standard version, where filtered projection data $Q(\rho, \theta)$ is mapped to its corresponding position within the $\mu(x, y)$ plane. Although, in the wavelet-based FBP, backprojection comprises the mapping of four different sets of filtered projection data, to create the four two-dimensional FWT coefficient images $W_s(k, l)$, $W_\psi^H(k, l)$, $W_\psi^V(k, l)$, and $W_\psi^D(k, l)$.
In the FBP, the backprojection is considered as the most demanding component within the reconstruction method, which for very large data sets, is even considered as a *bottleneck* [40]. In this section, it is once again worth

mentioning that by definition of the FWT, after a signal is mirror filtered, its bandwidth is reduced by half, so by means of the Shannon theorem, a FWT filtered signal can be represented with as half as many samples than the original signal, without compromising its integrity. For the two-dimensional FWT analysis it was shown in Chapter 3 that after being filtered the two-dimensional input signal is therefore subsampled along both rows and columns.
In the wavelet-based FBP, the subsampling operation of filtered data is part of the backprojection component, which means that instead of backprojecting filtered data over a full grid plane and subsampling performed to obtain FWT coefficient images, backprojection is directly executed over a reduced area pixel grid. Such pixel grid contain as half as many pixels than the input phantom, as shown below:

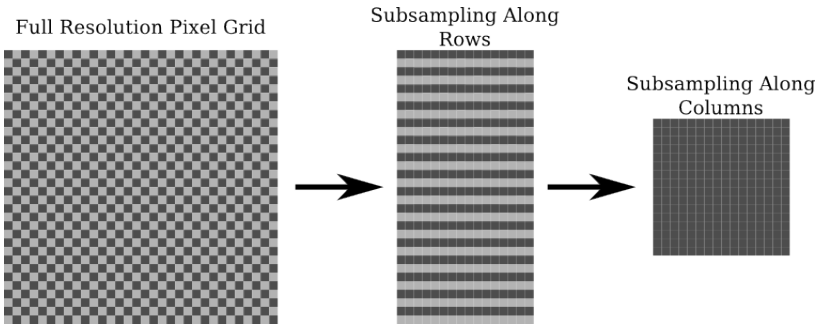


Figure 4.20: Pixel Grid Subsampling.

In the computer implementation, which is essentially the same as the one explained in Subsection 4.1.2, subsampling is accomplished during the creation of reference $\mathbf{X}$ and $\mathbf{Y}$, used to compute the $\mathbf{T}$ matrix that contain all displacement vector indexes. The computation of $\mathbf{T}$ that has been defined by Equation 4.19, is generalised to the following equation, by including subsampled reference matrices $\mathbf{X}$ and $\mathbf{Y}$.

$$\mathbf{T} = \mathbf{X} \downarrow_2 \cdot \cos\theta + \mathbf{Y} \downarrow_2 \cdot \sin\theta. \tag{4.28}$$

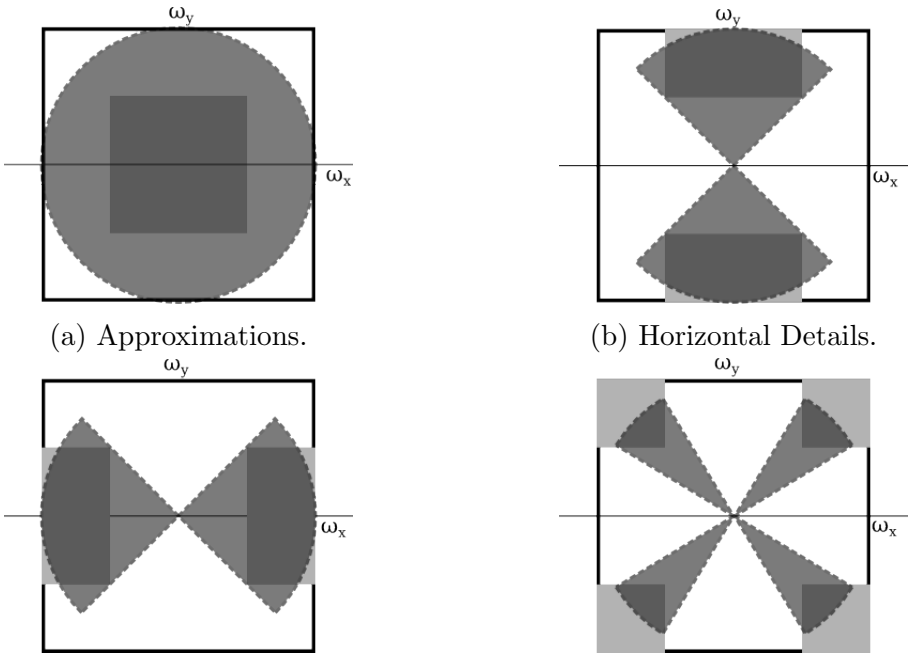
The modified MATLAB code for the creation of subsambled $\mathbf{X}$ and $\mathbf{Y}$ is simply achieved by incorporating a subsampling factor, that along with the parameters already defined in the previous code versions, the backprojection is manipulated to deliver images at the required resolution. For the example shown below, the subsampling factor has a value of 2, although for the multiresolution

representation that follows this section, such factor will be shown to be increased according the different scales.

```
%=========================================================================
%---X AND Y CARTESIAN COORDINATES ARRAYS AND MEMORY ALLOCATION---
%=========================================================================
% Define the x & y axes for the reconstructed image so that ...
    the origin
% (center) is in the spot which RADON would choose.
N = size(p,1);
center = floor((N+1)/2);
resolution = 2;  %subsampling factor
xleft = -center + 1;
x = (1:resolution:N) - 1 + xleft; %vector with -center:center ...
    in the x axis
x = repmat(x, N/resolution, 1);   %subsampled x matrix
ytop = center - 1;
y = (N:-resolution:1).' - N + ytop;
y = repmat(y, 1, N/resolution);   %subsampled y matrix
ctrIdx = ceil(len/2);     % index of the center of projections
t_proj = zeros(1,size(w_app,2));
%Memory allocation for coefficient images
imgd = zeros((N)/resolution,class(p));
imgv = zeros((N)/resolution,class(p));
imgh = zeros((N)/resolution,class(p));
imga = zeros((N)/resolution,class(p));
```

Another feature that arises from the inclusion of the two-dimensional FWT into the FBP method, and briefly mentioned in the previous subsection, is that according to the Fourier spectrum of high frequency filters (i.e. wavelet functions), it is possible to disregard the computation of a certain number of angular projections. This is illustrated in the next figure, where scaling and wavelet functions are shown in the Fourier space, along with the angular range in which angular projections is assumed to overlap with filter coefficients. In Figure 4.21, the support of the filters within the Fourier space is ideal, which means that high frequency filters are compactly supported and do not overlap between each other. In a real implementation, in order to achieve a set of compactly supported filters as the ones shown in Figure 4.21, it is necessary to employ a high order filter function constructed with a large amount of

coefficients. The latter can result disadvantageous, as employing a high order filter might be more demanding than disregarding a fewer amount of angular projections.



(a) Approximations. (b) Horizontal Details.

(c) Vertical Details. (d) Diagonal Details.

Figure 4.21: Fourier Space Overlapping Between Filter Coefficients and Angular Projections.

Coming up next, is the MATLAB implementation of the FWT coefficient images backprojection. Another important feature of the wavelet-based FBP algorithm, is that the backprojection of every FWT coefficient image is computationally independent from the others, which means that it is possible to generalise the wavelet-based FBP to a parallel computation framework.

```
%=========================================================================
%---BACKPROJECTION-------------------------------------------
%=========================================================================
%--------------approximations-------------
for i=1:length(theta) %backprojection involving all angles
    t_proj = w_app(i,:);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    imga = imga + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
```

```
    imagea = imga*(pi/(length(theta)));
end
%-----------------horizontal-------------
for i=13:169 %backprojection involving angles in the range from
    t_proj = w_hor(i,:);        %13 to 169, of 180
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    imgh = imgh + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imageh = imgh*(pi/(length(theta)));
end
%--------------vertical--------------------
for i=1:79  %1 to 9
    t_proj = w_vert(i,:);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    imgv = imgv + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imagev = imgv*(pi/(length(theta)));
end
for i=103:180   %103 to 180
    t_proj = w_vert(i,:);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    imgv = imgv + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imagev = imgv*(pi/(length(theta)));
end
%--------------diagonal--------------------
for i=8:83  %8 to 83
    t_proj = w_diag(i,:);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    imgd = imgd + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imaged = imgd*(pi/(length(theta)));
end
for i=99:174    %99 to 174
    t_proj = w_diag(i,:);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
```

```
47      imgd = imgd + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
48      imaged = imgd*(pi/(length(theta)));
49  end
```

Nonetheless, as it has been shown in the MATLAB example, the low order support-5 *daubechies* filter has been employed, it could be possible to disregard a certain amount of angular projections for the backprojection of details coefficient images. As can be seen in the graph shown in the next figure, the disregard of angular projections, made it possible to reduce the amount of operations involved in backprojection, so the execution time could be reduced. A comparison of the time spent to perform backprojection, between the standard FBP and the wavelet-based FBP, is presented below in Figure 4.22.

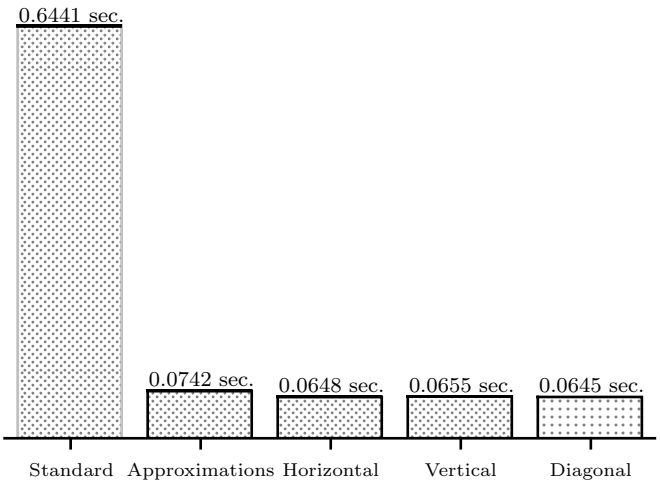
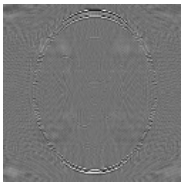


Figure 4.22: Backprojection Execution Time Comparison.

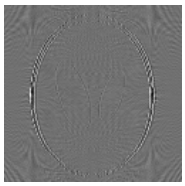
Such time measurements were taken by using the MATLAB 2015*a* *tic - toc* function, by using a computer with a $3.1GHz$ Intel Quad Core $i3$ processor and $4GB$ of RAM memory, running a 64-bit Fedora-17 distribution linux operative system. To ensure consistency in time measurements, a single processor core was employed and the average of ten, fifty and one hundred measurements were taken. This same time acquisition set-up is replicated along this chapter.
The next figure shows the reconstructed FWT coefficient images, that were obtained from the MATLAB implementation of the wavelet-based FBP algorithm. Up to this stage of the algorithm, it is not possible to make an evaluation of the reconstruction accuracy, although it can be seen from the

reconstructed images, that every coefficient match the expected frequency components: approximations containing the lower frequency components, and details containing the high frequency components along horizontal, vertical and diagonal directions.

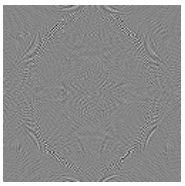
(a) Approximations Image.



(b) Horizontal Details Image.



(c) Vertical Details Image.



(d) Diagonal Details Image.

Figure 4.23: Reconstructed Two-Dimensional FWT Coefficient Images.

As these reconstructed lower resolution images are in the wavelet domain, it is necessary to calculate the inverse two-dimensional FWT, to obtain the final full resolution image that would be expected from a standard FBP reconstruction.

4.3.3 Two-Dimensional Inverse FWT of Reconstructed Coefficient Images

In contrast to the analysis two-dimensional FWT, that has to be modified in order to match with the format of the FBP framework, the synthesis counterpart is simply applied as its definition dictates it. That is, the inverse two-dimensional FWT consists in first upsampling the input coefficient images, approximations and details, along rows. Upsampled images are then reverse filtered along columns, and the results converged in the following manner: approximations are added with horizontal details, and vertical details are added with diagonal details. At this stage, input coefficient images have converged up to a couple of images, which now must be upsampled along columns. Upsampled images are reverse filtered along rows, and the results finally converged to yield the full resolution output image.
The synthesis process just described, is shown below in Figure 4.24, where every

applied operation to approximations and details input images, is illustrated along the computation of the synthesis framework.

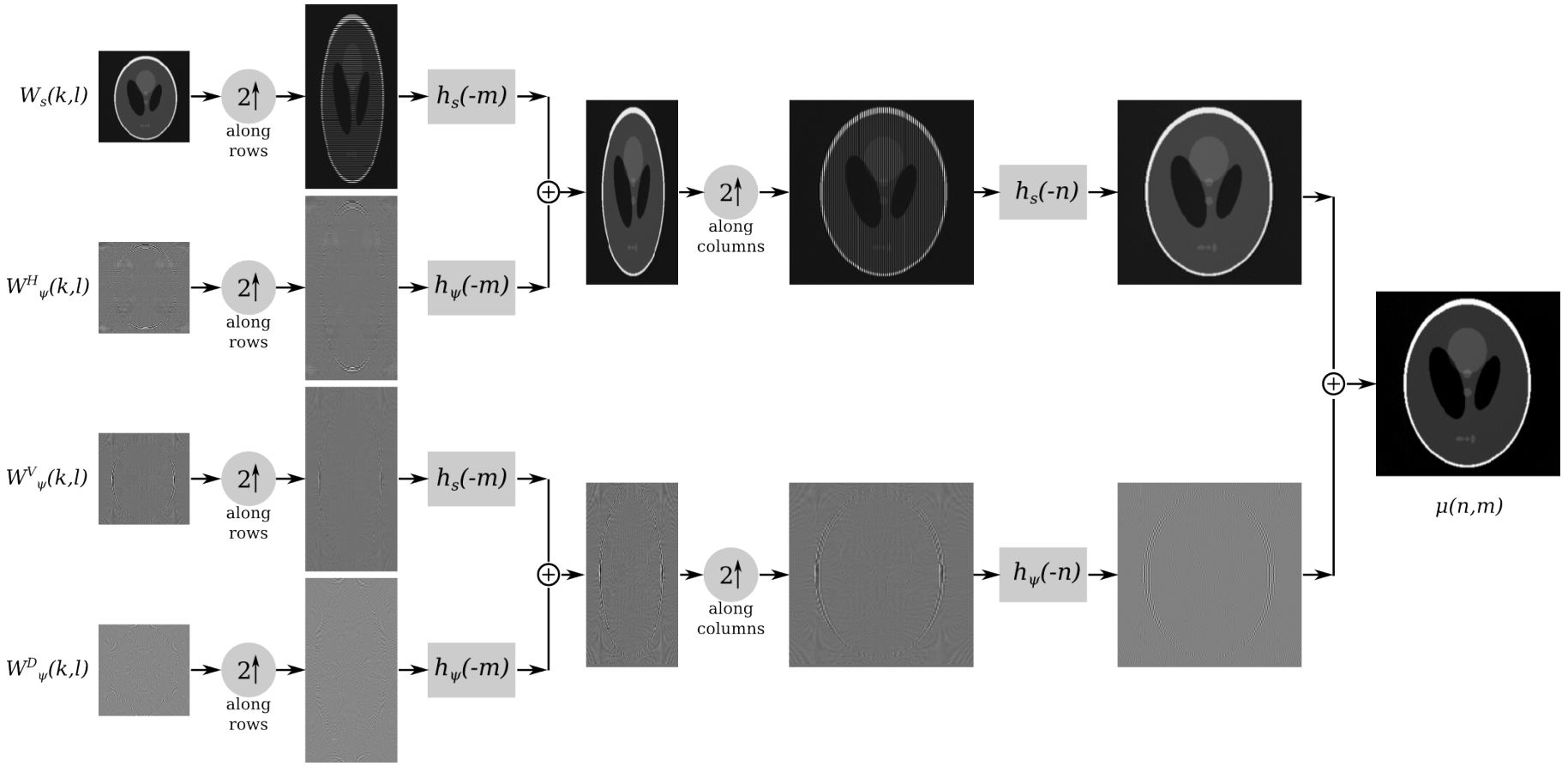
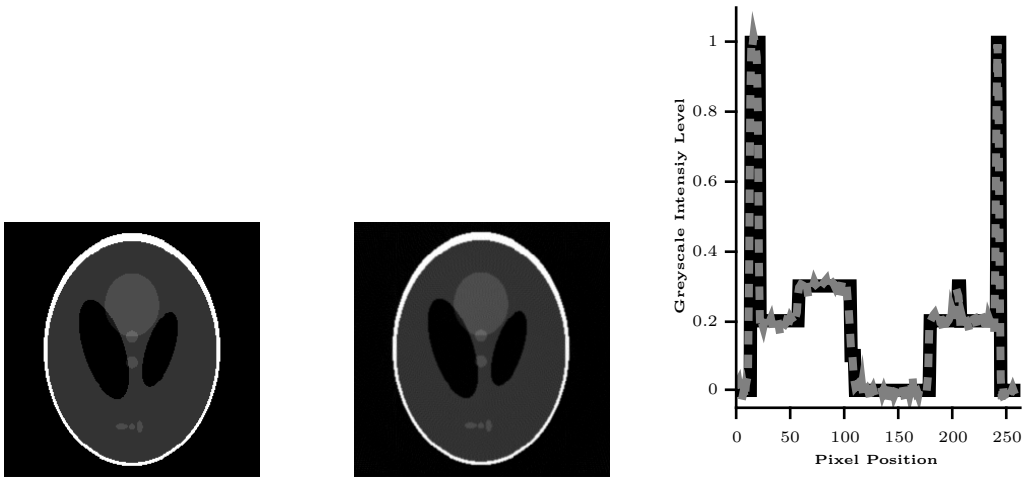


Figure 4.24: The Inverse Two-Dimensional FWT.

Images illustrating the operations involved in the inverse two-dimensional FWT, were obtained from the MATLAB implementation presented here:

```
%=====================================================================
%---WAVELET SYNTHESIS-------------------------------------
%=====================================================================
[lp, hp] = wavefilter('db5', 'r'); %Inverse filters
rdiag = upsample(imaged, 2); %coefficient images upsampling
rvert = upsample(imagev, 2); %along rows by a factor of two
rhor = upsample(imageh, 2);
rapp = upsample(imagea, 2);
rdiag = conv2(rdiag, hp.', 'same'); %reverse filtering
rvert = conv2(rvert, lp.', 'same'); %along columns
higher = rdiag + rvert;
rhor = conv2(rhor, hp.', 'same');
rapp = conv2(rapp, lp.', 'same');
lower = rhor + rapp;
higher = higher.';
lower = lower.';
higher = upsample(higher, 2); %upsampling along columns
lower = upsample(lower, 2);
higher = higher.';
lower = lower.';
t_lower = conv2(lower, lp, 'same'); %reverse filtering along
t_higher = conv2(higher, hp, 'same'); %rows
image = (t_higher + t_lower); %output full resolution image
```

After the inverse two-dimensional FWT of the reconstructed coefficient images has been achieved, it is possible to evaluate the accuracy of the final full resolution output image, which in the Figure 4.25 is compared with the input phantom. For the wavelet-based FBP image reconstruction example, the 256 by 256-pixel Shepp Logan phantom, has once again been employed.



(a) Input Phantom. (b) Reconstructed Image. (c) Greyscale Intensity Level along Row $y = 155$.

Figure 4.25: Comparison Between the Input Phantom and the Reconstructed Image.

By analysing the reconstructed image, against the input phantom, it can be implied that there are no distinctive differences, either in the intensity levels as well as the images structure. Although, as previously done in Section 4.2, the intensity levels of the pixel row $y = 155$, is plotted for both, the input phantom and the wavelet-based FBP reconstructed image.

In the plot shown in Figure 4.25c, the black line belongs to the input phantom, while the grey dotted line, correspond to the reconstructed image. From such comparison, it can be shown that the wavelet-based FBP fairly preserves the same structure of the input phantom.

Again, as applied to the reconstruction results shown in previous sections, numerical evaluation of the reconstructed image quality is shown in the next table by using the same metrics explained in Subsection 4.1.1. The table include the error calculated for the reconstruction obtained through the standard FBP, calculated in Section 4.2, and the wavelet-based approach explained in this Section. Both by taking the input phantom as reference image.

Method	AVERR	NABS	MSE	PSN	SSIM
Standard	0.0174	0.1419	0.0018	27.3788	0.8663
Wavelet	0.0288	0.2345	0.0111	19.5354	0.7533

Table 4.4: Reconstruction Quality Comparison of the Full and Half Frequency Spectrum Image Reconstructions.

From table 4.4, it can be seen that the inclusion of the FWT in the FBP involve a small degradation of the reconstructed image, compared to the standard FBP. It is appropriate to remark that, although the backprojection time of the FWT coefficient images resulted to be considerable smaller than the standard FBP version, as shown in Figure 4.22, the execution time of the inverse two-dimensional FWT was not considered. By using the same methodology, the measured time of the FWT synthesis resulted in an approximate value of 0.0048 seconds, therefore by updating the synthesis time in Figure 4.22, Figure 4.26 displays the overall time needed to backproject each of the two-dimensional FWT coefficient images, including the synthesis time:

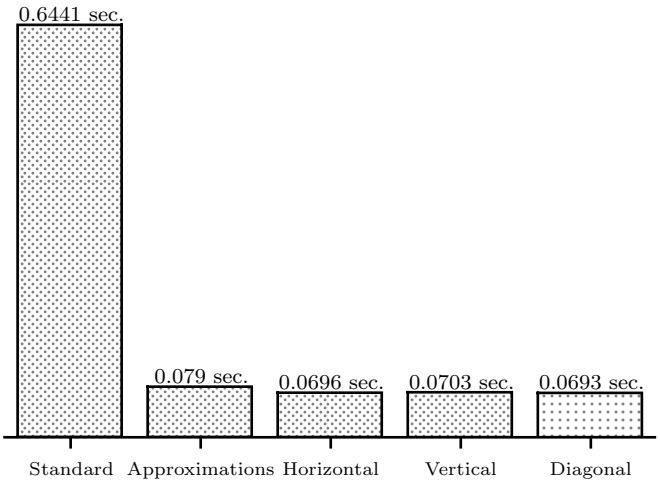


Figure 4.26: Backprojection Execution Time Comparison.

Through this simple example, it was possible to demonstrate that the inclusion of the two-dimensional FWT into the FBP affect the accuracy of the reconstructed image, although it is around eight times faster when assuming a parallel image reconstruction of every FWT coefficient image. Therefore it can be concluded that the speed gain provided by the FWT comes at the expense of small degradation of the reconstructed image.

4.4 Wavelet-based, Parallel Multiresolution FBP

In the previous section, it has been explained how the inclusion of the two-dimensional FWT can provide some improvements to the FBP image reconstruction method. The simplification of the backprojection computation, covered in Subsection 4.3.2, is considered the most notable. The other highlighted benefit was the allowance to disregard a certain amount angular projections to simplify the computation of details coefficient images. Although the latter mentioned benefit proved to reduce the reconstruction complexity, it is exclusive to the details coefficients and can not be applied to the approximations. Such a statement can be verified by looking at the Fourier space tiling of the scaling function in Figure 4.17a, which shows how it overlaps all angular projections.
As will be seen in this section, the two-dimensional FWT decomposition of projection data, not only into frequency components but also at different scales, can improve the reconstruction of the approximations coefficient image. Moreover, the multiresolution decomposition is a crucial element for the achievement of what drove this proposed algorithm to employ the two-dimensional FWT: the fast and accurate reconstruction of ROIs, from truncated projection data.

4.4.1 Multiresolution Decomposition of Projection Data

As a reminiscence of what was exposed in Section 3.5, for a multiresolution representation of an image through the analysis of the two-dimensional FWT, it is required that such image be iteratively decomposed into a set of two-dimensional coefficients, having half as many row and column samples than its nearest higher resolution image coefficient. At each iteration, the output details coefficients are saved and approximations used as the input for a new scale decomposition. Each iteration is equivalent to a scale depth, so for a given number of iterations, details coefficients are obtained; conversely, approximations coefficient, is only obtained from the latest iteration (i.e. lower resolution). The analysis the two-dimensional FWT for two scales decomposition is shown below.

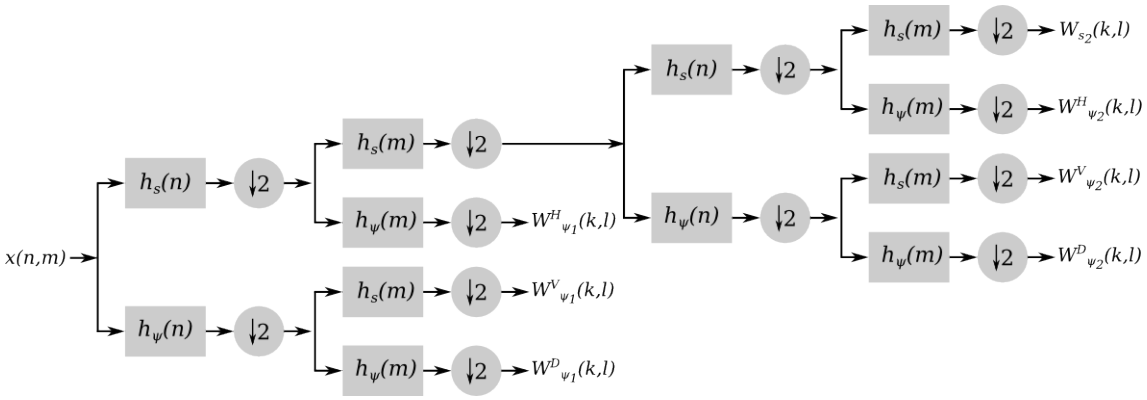


Figure 4.27: Two-Scale Multiresolution Analysis Through Standard Two-Dimensional FWT.

Applied to Tomography, the multiresolution two-dimensional FWT can improve the FBP reconstruction, in the sense that at each decomposition iteration, coefficient images are subsampled along rows and columns by a factor of two. This means that, for projection data that has been filtered with a wavelet ramp filter, backprojection should be performed to reduced area pixel grids, getting smaller at each scale. The next figure shows the two scale tiling of the Fourier space, from which it can be seen that although all angular projections must be considered in the approximations coefficient (denoted by $\mathbf{A_2}$) reconstruction, backprojection must be performed to a pixel area that is one eight of the full resolution image size.

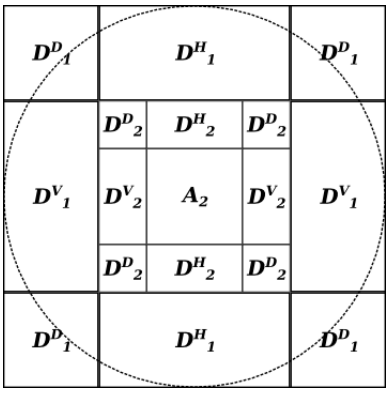


Figure 4.28: Fourier Space Tiling by the Two-Dimensional FWT.

Although the overall simplification of the FWT coefficient images reconstruction seems to be achieved by the multiresolution representation, it must be remarked that for the wavelet-based FBP reconstruction, either for a single or multiple

resolutions, the complexity reduction is valid only when the reconstruction of coefficient images is performed in parallel. Unfortunately, such feature do not match with the scheme of the multiresolution FWT analysis, where details coefficients at lower scales, are recursively obtained from higher resolution approximations.

4.4.2 Parallel Multiresolution Decomposition of Projection Data

In [141] Tay and Havlicek, with the purpose to achieve an orientation selective two-dimensional FWT analysis filter bank, they experienced with the direct implementation of the FWT in the Fourier domain. Such implementation exhibited that subsampling could also be performed in the Fourier domain, so if applied in the wavelet-based FBP, it could be possible to avoid the need to perform the backprojection of the coefficient images, before applying subsampling, in the case of a multiresolution decomposition.
According to [142, p.91], the subsampling of a Fourier domain signal $X(\omega)$ by a factor of two, can be simply performed through:

$$V(\omega) = \frac{1}{2}\left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} + \pi\right)\right], \tag{4.29}$$

which for an computer generated FFT sequence $X(k)$, with length N, it follows:

$$V(k) = \frac{1}{2}\left[X(k) + X\left(\frac{N}{2} + k\right)\right]. \tag{4.30}$$

Equation 4.30 clearly accomplishes to avoid the recursive backprojection of approximations coefficient, by allowing the subsampling in the same format in which projection data is filtered, just before being translated back to the Radon space domain, in order to be backprojected. Nonetheless, generating filtered projection data at different scales still a recursive operation. Because of that reason, subsampling in the Fourier domain results insufficiently in the pursuit to achieve a true parallel implementation of the multiresolution wavelet-based FBP.
In [142, p.100], it is explained how a multiresolution filter bank analysis, in its standard form, is somehow inefficient in the sense in that subsampling follows filtering. In other words, through a cascaded filter bank, a signal is first filtered

and then many of its calculated components thrown away by the subsampling. Reversing the order of such operations, either for subsampling (for the analysis) or upsampling (for the synthesis), can be done according to the Noble identities [141], for the Fourier implementation of multiresolution filter banks. For the analysis case, the Noble identities state that subsampling by M and then filtering by $G(z)$, is equivalent to filtering by $G(z^M)$ and subsampling by M.

Figure 4.29: Subsampling Noble Identity.

For the synthesis case, filtering by $G(z)$ and then upsampling by M, is equivalent to upsampling by M and then filtering by $G(z^M)$.

Figure 4.30: Upsampling Noble Identity.

By making use of the Noble identities, it is therefore possible to adapt the recursive problem to a parallel scheme. This is shown in the next example, where the analysis filters for a three scale decomposition one dimensional filter bank, are calculated. By preserving the notation shown in [141], analysis filters $H(z)$ for scales 1, 2, and 3 are calculated from low pass $F_a(z)$ and high pass $G_a(z)$ Fourier domain filters.

$$\begin{aligned} H_3(z) &= F_a(z)F_a(z^2)G_a(z^4) \\ H_2(z) &= F_a(z)G_a(z^4) \\ H_1(z) &= G_a(z) \end{aligned} \tag{4.31}$$

Synthesis filters $\tilde{H}(z)$ for each scale, are similarly calculated from reverse low pass and high pass filters in Fourier domain, $F_s(z)$ and $G_s(z)$.

$$\begin{aligned} \tilde{H}_3(z) &= G_s(z^4)F_s(z^2)F_s(z) \\ \tilde{H}_2(z) &= G_s(z^2)F_s(z) \\ \tilde{H}_1(z) &= G_s(z) \end{aligned} \tag{4.32}$$

The equivalent one-dimensional filter bank, for both the analysis and synthesis, is therefore constructed as follows:

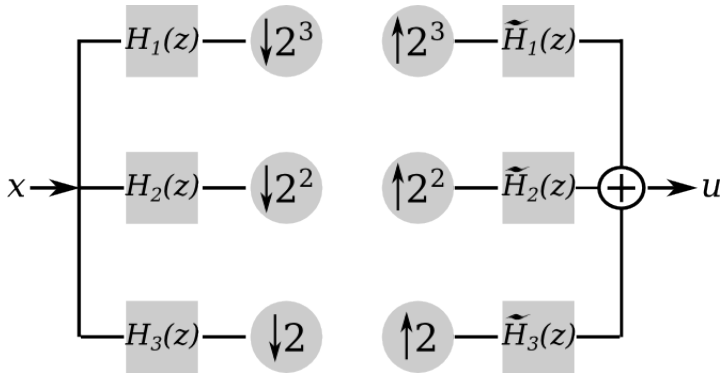


Figure 4.31: Three Scale Parallel Filter Bank [141].

By adapting the mathematical expressions, of the filter calculations shown in Equations 4.31 and 4.32, to the notation used along this thesis, it is then possible to express the calculation of multiresolution analysis filters $H_s(u)$ at a specific scale l through:

$$h_{s_l}(n) \leftrightarrow H_{s_l}(u) = \begin{cases} \prod_{q=0}^{l-1} H_s(2^q u) & l = 1, 2, 3, \ldots \\ 1 & l = 0 \end{cases} \tag{4.33}$$

And similarly for the synthesis filters $H_\psi(u)$, that yield to:

$$h_{\psi_l}(n) \leftrightarrow H_{\psi_l}(u) = \begin{cases} H_\psi(2^{l-1}u) H_{s_{l-1}}(u) & l = 1, 2, 3, \ldots \\ 1 & l = 0 \end{cases} \tag{4.34}$$

Through this set of multiresolution filters, it is then possible to construct the two-dimensional filters, that would allow to decompose analysis projection data into different scales, where the backprojection of details and approximations coefficient images can be performed in parallel. It is important to mention that Equations 4.33 and 4.34 were taken from [33], where the authors indicate its utilisation to obtain what they define as the *nonpyramidal* wavelet decomposition.

The MATLAB implementation of the multiresolution analysis filters for a three scale decomposition was done by using the support-5 *Biorthogonal 2.4* wavelet and scaling filters. As can be seen below, the calculation of such filters is very

simple, and although their coefficients are calculated recursively, they are not necessarily required to be calculated during runtime. This way, for a parallel implementation, filter coefficients can be pre-calculated and stored in memory, avoiding to interfere with the parallel execution of the algorithm. To avoid redundancy in the MATLAB code implementation shown below, the FFT of the filter *Biorthogonal 2.4*, is indicated through the function *ft_wavefilter*, which correspond also to the scale $l = 1$ coefficients.

```
[lp hp] = wavefilter('bior2.4', 'd');
%=========================================================================
%l=1 Filters
%=========================================================================
[f_lp f_hp] = (ft_wavefilter(lp, hp, order)); %FFT of lp and hp
%=========================================================================
%l=2 Filters
%=========================================================================
f_lp_2 = zeros(1,length(f_lp));
f_hp_2 = zeros(1,length(f_lp));
f_lp_m = [f_lp f_lp];
f_hp_m = [f_hp f_hp];
for i =1:length(f_lp)
    f_lp_2(i) = (f_lp(i).*f_lp_m(2*i)); %l=2 scaling function
    f_hp_2(i) = (f_hp_m(2*i).*f_lp(i)); %l=2 wavelet function
end
%=========================================================================
%l=3 Filters
%=========================================================================
f_lp_3 = zeros(1,size(f_lp,2));
f_hp_3 = zeros(1,size(f_lp,2));
f_lp_m_3 = [f_lp_m f_lp_m];
f_hp_m_3 = [f_hp_m f_hp_m];
for i =1:length(f_lp)
    f_lp_3(i) = (f_lp_2(i).*f_lp_m_3(4*i)); %l=3 scaling function
    f_hp_3(i) = (f_hp_m_3(4*i).*f_lp_2(i)); %l=3 wavelet function
end
```

The three scale *Biorthogonal 2.5* low pass and high pass filters obtained from the MATLAB implementation are shown in Figure 4.32, where the thick black dotted curves correspond to scale one filters, which are simply obtained through the FFT of the impulse responses *Biorthogonal 2.5* coefficients. Grey thick and

thin curves, correspond to scales two and three low pass and high pass filters, which were calculated according to Equation 4.33.

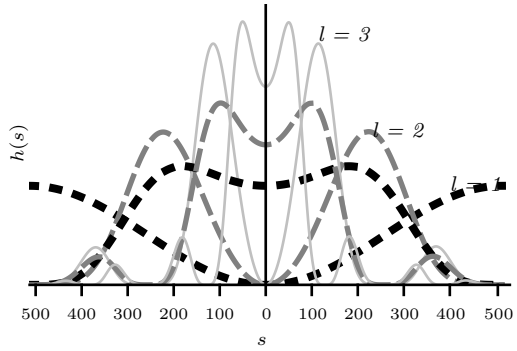
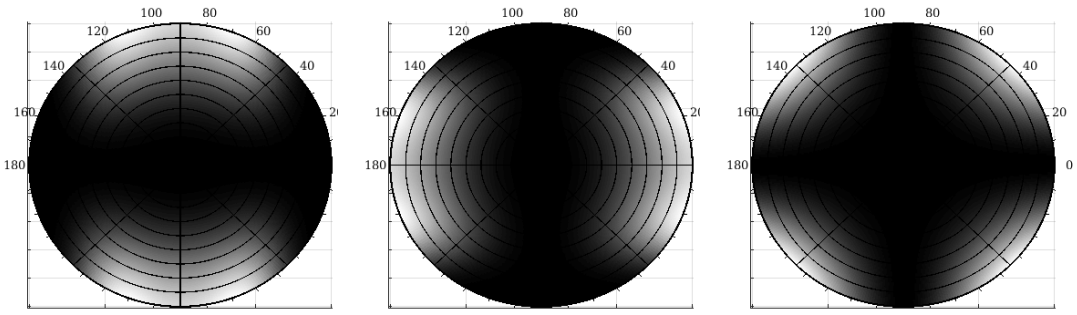


Figure 4.32: Multiresolution Low Pass and High Pass *Biorthogonal 2.4* Fourier Domain Filters.

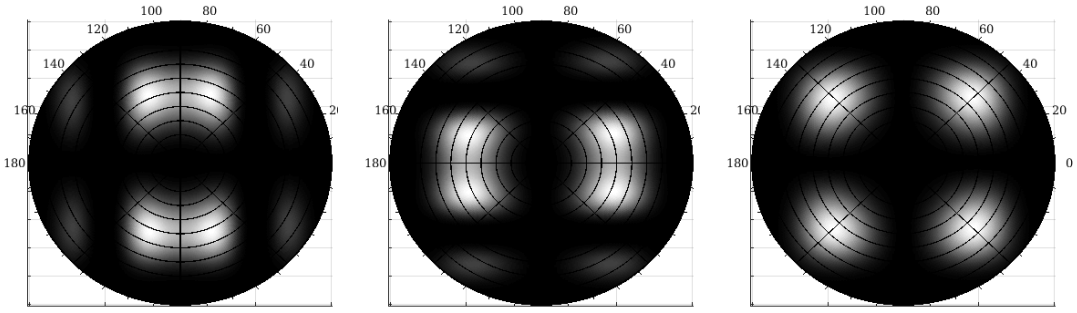
Analogous to what was previously done with low pass and high pass filters, in the construction of the wavelet-based FBP algorithm, multiresolution filters are also required to be translated to the polar coordinate system. Similarly, the separable products to yield the corresponding two-dimensional wavelet and scaling functions, for the multiresolution scheme, is as well accomplished through Equations 4.22 and 4.22.
As stated previously, the scaling two-dimensional filter, to yield the approximations coefficient image, is only required for the lower resolution. Therefore only scale-1 *Biorthogonal 2.5* wavelet two-dimensional filters are required.



(a) Horizontal. (b) Vertical. (c) Diagonal.

Figure 4.33: Scale-1 Two-dimensional Wavelet Analysis Filters.

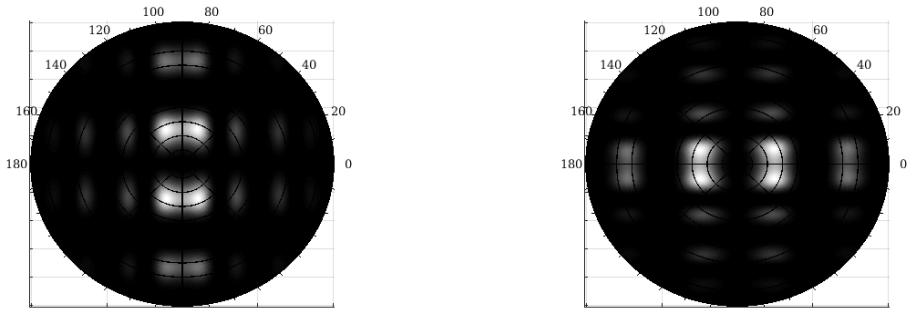
The same assumption applies for the scale-2 wavelet two-dimensional filters.



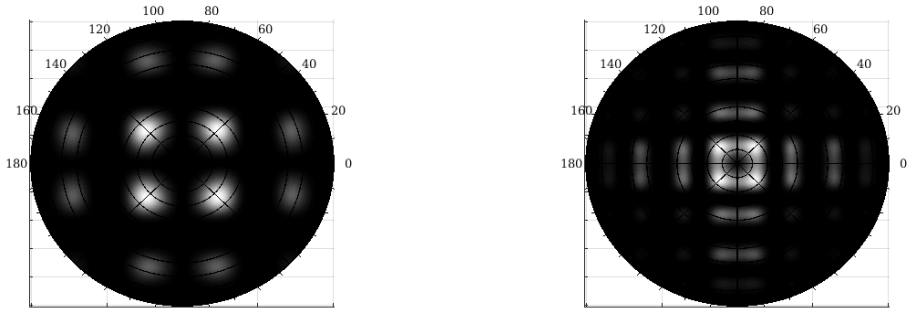
(a) Horizontal. (b) Vertical. (c) Diagonal.

Figure 4.34: Scale-2 Two-dimensional Wavelet Analysis Filters.

For the lower resolution (scale-3 in this example), the scaling function to obtain the approximations coefficient image, is also calculated.



(a) Horizontal. (b) Vertical.



(c) Diagonal. (d) Approximations.

Figure 4.35: Scale-3 Two-dimensional Wavelet and Scaling Analysis Filters.

Figures 4.33, 4.34 and 4.35, show through a 3-scale decomposition example, how the Fourier space is tiled by the multiresolution wavelet and scaling filters. From such Fourier space tiling, it is then possible to realise that a true parallel multiresolution analysis is achievable. This way, the problem of decomposing projection data into a set of frequency components, belonging to different scales, and in parallel, can be achieved through the application of Equations 4.33 and 4.34, derived from the Noble identities.
Applied to the Tomography reconstruction problem, multiresolution analysis filters are incorporated to the FBP, in a similar way as they were included in the wavelet-based FBP. The only difference is that, for the multiresolution parallel scheme, the number of filtering components is increased according to the number of approximations and details coefficient images, which depends upon the chosen scale. The MATLAB code shown below, is a 3-scale decomposition example, where projection data in Fourier domain and denoted by the variable *ft_sngrm*, is decomposed into nine details coefficients and one approximations.

```
%==================================================================
%---MULTIRESOLUTION WAVELET RAMP FILTERING IN FOURIER DOMAIN----
%==================================================================
hor_filt = ml.*nh.*ramp_filt.*ft_sngrm;  %scale-1 filtered data
vert_filt = mh.*nl.*ramp_filt.*ft_sngrm;
diag_filt = mh.*nh.*ramp_filt.*ft_sngrm;

hor_filt_2 = ml_2.*nh_2.*ramp_filt.*ft_sngrm; %scale-2
vert_filt_2 = mh_2.*nl_2.*ramp_filt.*ft_sngrm;
diag_filt_2 = mh_2.*nh_2.*ramp_filt.*ft_sngrm;

hor_filt_3 = ml_3.*nh_3.*ramp_filt.*ft_sngrm; %scale-3
vert_filt_3 = mh_3.*nl_3.*ramp_filt.*ft_sngrm;
diag_filt_3 = mh_3.*nh_3.*ramp_filt.*ft_sngrm;
app_filt_3 = ml_3.*nl_3.*ramp_filt.*ft_sngrm;
```

Such difference is better illustrated in Figure 4.36, through the structure of the parallel multiresolution FWT analysis filter bank, for the FBP reconstruction of approximations and details coefficient images, $W_{s_l}(k,l)$ and $W_{\psi_l}(k,l)$. In such a filter bank, synthesis filters are as well indicated in terms of the involved product to create the wavelet modified ramp filters $\mathcal{W}_{\psi_l}$ and $\mathcal{W}_{s_l}$.

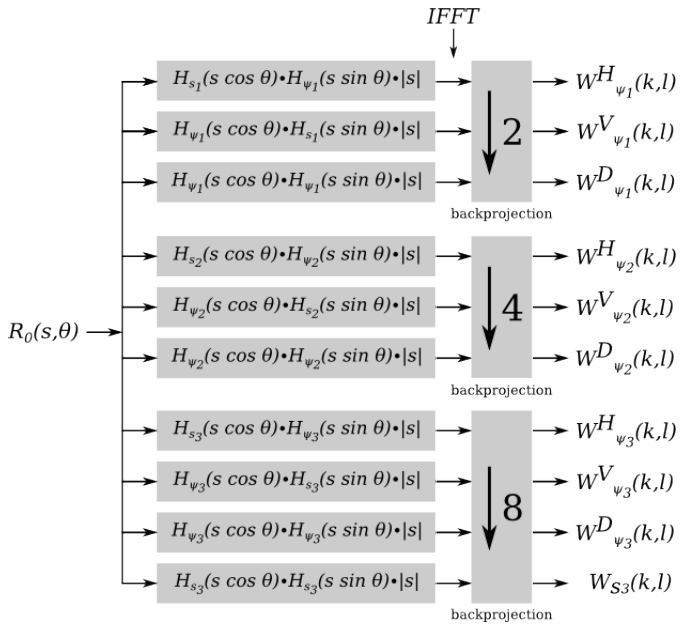


Figure 4.36: Parallel Multiresolution Analysis Filter Bank.

From the same figure, it is possible to see that not only the filtering component involve some changes, but also the backrpojection. In the parallel multiresolution analysis, the subsampling is performed according to the scale of filtered data. For every scale, subsampling increases by a factor of two, so for the 3-scale example, subsampling keeps one sample out of two in $l = 1$, one out of four in $l = 2$ and one out eight in $l = 3$. If once again, the 256 by 256-pixel Shepp Logan phantom is employed, coefficient images at $l = 1$ will be formed by 128 by 128 pixels, for $l = 2$ 64 by 64-pixel images will be obtained, and 32 by 32-pixel images for $l = 3$.
To illustrate such differences in the MATLAB computer implementation, the $l = 2$ allocation of the arrays that define the Cartesian grid, to which the backprojection operator maps filtered projection data to form the output coefficient images, is given by:

```
%=================================================================
%---X AND Y CARTESIAN ARRAYS AND MEMORY ALLOCATION FOR l=2---
%=================================================================
resolution = 4; %scale 2
x_2 = (1:resolution:N) - 1 + xleft;
x_2 = repmat(x_2, (N/resolution), 1);
y_2 = (N:-resolution:1).' - N + ytop;
```

```
y_2 = repmat(y_2, 1, (N/resolution));
%Memory Allocation
imgd_2 = zeros((N/resolution),class(p));
imgv_2 = zeros((N/resolution),class(p));
imgh_2 = zeros((N/resolution),class(p));
```

where the variable *resolution* indicates the interval at which indexes extract the corresponding Radon domain intensity values to form output images. Arrays denoted by the variables *imgd_2*, *imgv_2*, and *imgh_2*, are the allocated memory grids of size 64 by 64, employed to hold the pixels of the output diagonal, vertical, and horizontal, details coefficients images. The backprojection component for the multiresolution reconstruction, is implemented under the same principles of the standard FBP method. For this reason, the corresponding MATLAB code has not been included here. Although, it is enough to indicate that the only difference with other explained FBP versions, is the handling of the allocated memory elements defined according to the size of each of the output coefficient images. More detail on backprojection computer implementation has been given in Subsection 4.1.2.
To better illustrate the efficiency of the multiresolution, wavelet-based FBP reconstruction, approximations, as well as details output coefficient images, are brought together into Figure 4.37. Such results were obtained through the MATLAB implementation of the parallel filter bank in Figure 4.36, which confirms what was discussed at the beginning of this section; a parallel multiresolution implementation allows to reduce the computation of the approximations coefficient image, shown at the top left corner of Figure 4.37.

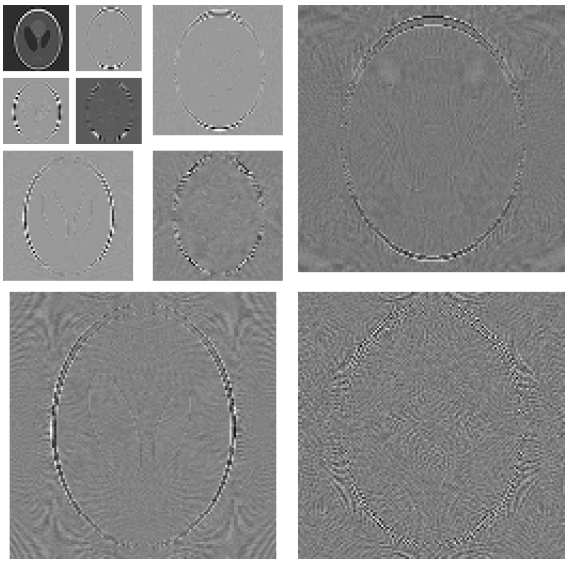


Figure 4.37: Multiresolution Approximations and Details Coefficient Images.

To achieve the reconstruction of the full resolution output image (i.e. scale $l = 0$), a synthesis filter bank is employed. As in the analysis counterpart, the synthesis can also by adapted to a parallel scheme as is explained in the next subsection.

4.4.3 Parallel Multiresolution Synthesis of Reconstructed Coefficient Images

The synthesis counterpart of the parallel FWT analysis is achieved in a very similar way, where the upsample Noble identities are employed. As it was shown in Figure 4.31, the synthesis parallel scheme is simply constructed as a reverse version of the analysis version. In the synthesis, downsamplers are substituted by upsamplers, and multiresolution filters by reverse multiresolution filters. Applied to the synthesis of the multiresolution wavelet-based FBP reconstructed coefficient images, the parallel filter bank of Figure 4.31 is not applicable. The reason why such an approach does not accomplish the required inversion of the coefficient images, relies in the domain of the inversion filters, which as being derived from the Noble identities, are calculated in Fourier domain [142, p.91]. This feature does not represent a real limitation, and the appropriate multiresolution synthesis filters can be calculated to be applied in the required spatial domain.

By calculating multiresolution synthesis filters in the appropiate domain, the parallel filter bank for the two-dimensional FWT synthesis of reconstructed, approximations and details coefficient images, is presented in the next figure.

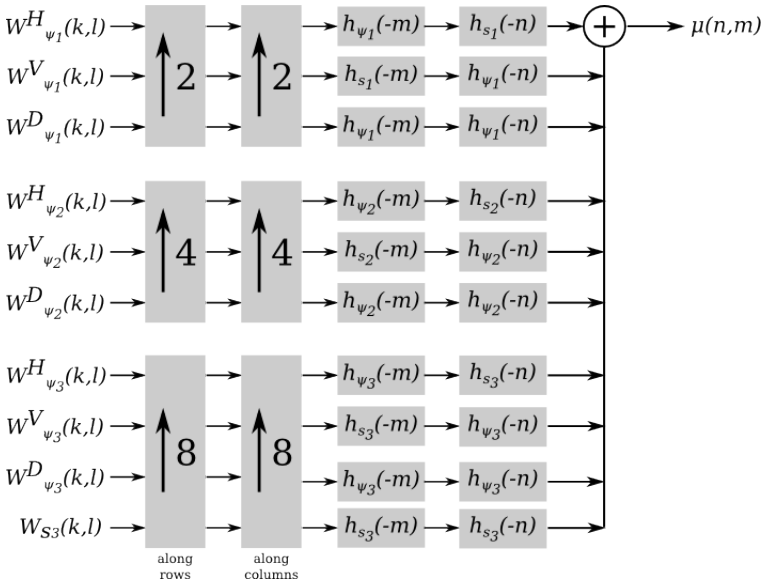


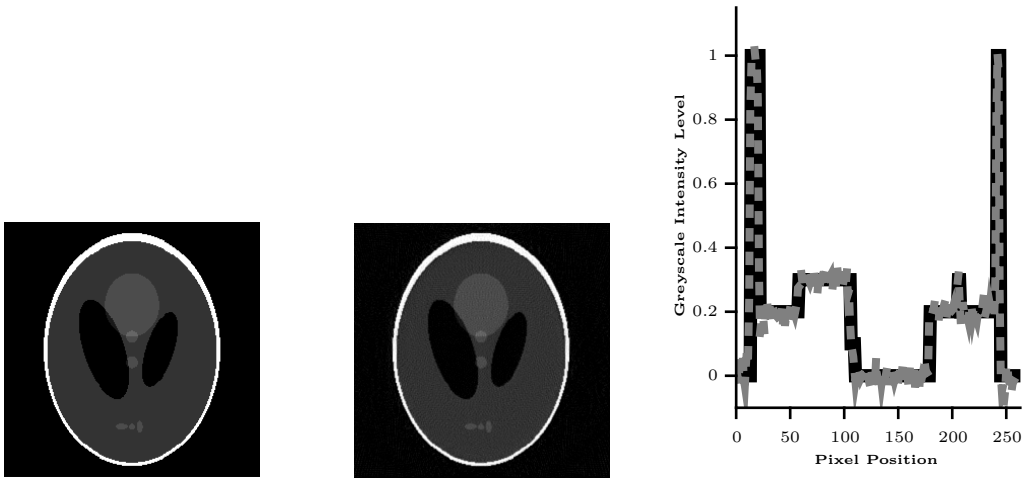
Figure 4.38: Parallel Multiresolution Synthesis Filter Bank.

Where $W^H_{\psi_l}(k,l)$, $W^V_{\psi_l}(k,l)$, and $W^D_{\psi_l}(k,l)$, are the horizontal, vertical, and diagonal details coefficient images. W_{s_3} is the approximations coefficient image, at the lower scale 3. Upsamplers $\uparrow 2$, $\uparrow 4$, and $\uparrow 8$, are applied to the rows and columns of the input coefficients, before being reverse filtered. Reverse filtering in the spatial domain, is then performed through the convolution between the pair of low pass h_{s_l} and high pass h_{ψ_l} filters, along columns and rows of the upsampled coefficient images. Finally, all the reverse filtered elements are merged to yield the full resolution output image.
To explain the MATLAB implementation of the parallel multiresolution, two-dimensional FWT, the piece of code that belongs to the computation of the scale-3 details and approximations images synthesis is presented below. In such code sample, *lp* and *hp* are the low and high pass filters, calculated for the scale 3. Such filters were obtained through the application of the Noble identities in Fourier domain, and brought back to its original domain, to be used as a sequence of scale-3 *Biorthogonal2.4* wavelet and scaling filter coefficients.

```
1 %==============================================================
2 %---l=3 WAVELET SYNTHESIS-------------------------------------
```

```
%=====================================================================
lp = t_ls_3;    %Scale 3 low-pass and high-pass filters
hp = t_hs_3;
rdiag_3 = upsample((imgda_3), 8, 3); %upsampling along rows
rvert_3 = upsample((imgva_3), 8, 3); %scale 3 equals to 1/8 ...
rhor_3 = upsample((imgha_3), 8, 3);  %of scale 0
rapp_3 = upsample(imgaa_3, 8, 3);
rdiag_3 = rdiag_3.';
rvert_3 = rvert_3.';
rhor_3 = rhor_3.';
rapp_3 = rapp_3.';
rdiag_3 = upsample(rdiag_3, 8, 3); %upsampling along columns
rvert_3 = upsample(rvert_3, 8, 3);
rhor_3 = upsample(rhor_3, 8, 3);
rapp_3 = upsample(rapp_3, 8, 3);
rdiag_3 = rdiag_3.';
rvert_3 = rvert_3.';
rhor_3 = rhor_3.';
rapp_3 = rapp_3.';
rdiag_3 = conv2(rdiag_3, hp.', 'same'); %filtering across columns
rvert_3 = conv2(rvert_3, lp.', 'same');
rhor_3 = conv2(rhor_3, hp.', 'same');
rapp_3 = conv2(rapp_3, lp.', 'same');
rdiag_3 = conv2(rdiag_3, hp, 'same');   %filtering across rows
rvert_3 = conv2(rvert_3, hp, 'same');
rhor_3 = conv2(rhor_3, lp, 'same');
rapp_3 = conv2(rapp_3, lp, 'same');
image_3 = (rapp_3 + rdiag_3 + rvert_3 + rhor_3); %convergence ...
    of filtered coefficients
```

The same code is replicated for scales 1 and 2, except for the synthesis of the approximations coefficient image, which is exclusive of the lower resolution. The example to visualise the efficiency of the multiresolution wavelet-based FBP, started in Subsection 4.4.2, is complemented through the parallel two-dimensional FWT synthesis of the reconstructed details and approximations coefficient images of Figure 4.37. The full resolution output and the input phantom, are shown in Figure 4.39, along with the plot of the pixel row located in $y = 113$, to show the correspondence of grey scale intensity levels, as well as the preservation of the image's pixel structure.



(a) Input Phantom. (b) Multiresolution Reconstructed Image. (c) Greyscale Intensity along Row $y = 155$.

Figure 4.39: Comparison Between the Input Phantom and the Multiresolution Reconstructed Image.

Again, as shown in Figure 4.39c, it is possible to see that the reconstructed image stills fairly preserving the structure of the input phantom, no matter that it has been obtained from a multiresolution decomposition.
To provide more sense about the quality of the reconstructed image, the following table shows the error values obtained from the standard, wavelet-based and the multiresolution versions of the FBP. All of the error metrics have again been calculated by considering the input phantom as reference image.

Method	AVERR	NABS	MSE	PSN	SSIM
Standard	0.0174	0.1419	0.0018	27.3788	0.8663
Wavelet	0.0288	0.2345	0.0111	19.5354	0.7533
Multiresolution	0.0287	0.2339	0.0109	19.6235	0.7706

Table 4.5: Reconstruction Quality Comparison of the Full and Half Frequency Spectrum Image Reconstructions.

Surprisingly, the multiresolution approach performed slightly better than the single resolution wavelet-based version of the FBP, in terms of quality and with around the same speed gain.
It was stated at the beginning of this section, that through the parallel implementation of a multiresolution decomposition, the complexity reduction of the approximations output coefficient should be achieved as well. The next

figure shows a graph displaying the reconstruction time that each of the details and approximations coefficient images, spent along with the parallel two-dimensional FWT synthesis. As expected, the overall time was considerably reduced, as compared with the standard FBP reconstruction.

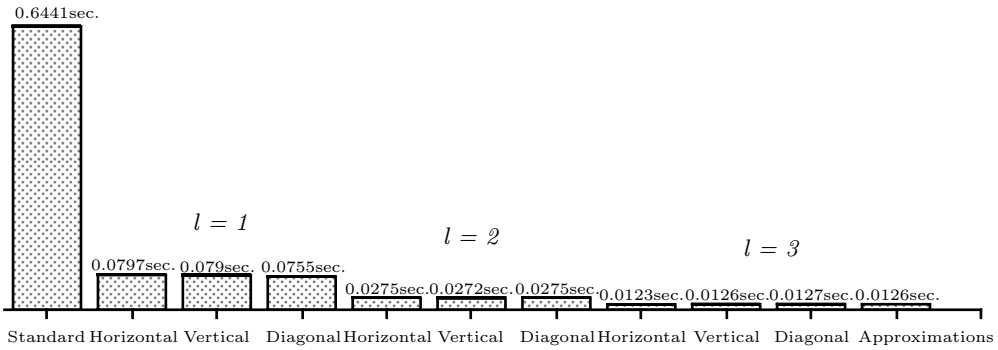


Figure 4.40: Wavelet-based, Parallel Multiresolution Reconstruction Times.

A very similar approach, in which tomographic projection data is decomposed into frequency components prior to perform image reconstruction, has been proposed by Thomas Rodet and Laurent Desbat in [41]. In Rodet-Desbat's approach, functions that tile the Fourier space into a certain number of adjacent squares, are calculated to be included in the filtering component of the FBP. Through such functions, they decompose projection data into what they call *frequency channels*. From every frequency channel, an image is reconstructed through the normal backprojection operator, and subsequently merged into the full size image. The gain in speed of Rodet-Desbat's approach, resides in the allowance to discard the angular projections that do not contribute to each of the frequency-channel images, according to the tiling of the Fourier space. Differently to the parallel multiresolution decomposition approach presented in this thesis, Rodet-Desbat's work does not consider the subsampling of frequency channels during the backprojection, so no matter into how many frequency channels projection data is decomposed, the backprojection is always performed over a full size pixel-grid.
The parallel multiresolution wavelet-based FBP algorithm, not only allowed to reduce the reconstruction time, but also to achieve the appropriate framework that led us to achieve the parallel block reconstruction algorithm that is presented in the next section, and represents the major contribution of this thesis.

Before concluding this section, it is appropriate to mention why the support-5 *Daubechies* filters were substituted by the *Biorthogonal2.4*, of the same support. Both of the filters are constructed with the same number of coefficients, but with the difference that *Daubechies* basis functions are asymmetric.
The implication of such characteristic relies in the calculation of the synthesis filters for the multiresolution decomposition, which are first calculated in Fourier domain according to the Noble identities and then translated to the space domain to match with the reconstruction algorithm requirements. Therefore, the appropriate symmetry of synthesis filters in space domain is determinant in the accuracy of the reconstructed image, and can be easily achieved by employing wavelet filters that are symmetric in nature as it is the case for the *Biorthogonal* basis functions.
For the requirements of this application, symmetry was fulfilled by employing *Biorthogonal* basis functions. Figure 4.41 shows the reconstruction results from three different wavelet basis functions: *Daubechies5*, *Symlets5* and the *Biorthogonal2.4*. All of them having the same support.
Table 4.6 show the numerical values of the calculated error from the reconstruction performed with every of the mentioned different wavelet basis. Results clearly show that symmetric wavelet perform better.

(a) Asymmetric *Daubechies5*.

(b) Near Symmetric *Symlets5*.

(c) Symmetric *Biorthogonal2.4*.

Figure 4.41: Comparison Between Reconstruction Accuracy by Employing Different Wavelet Basis with Different Symmetry.

Wavelet Basis	AVERR	NABS	MSE	PSN	SSIM
Daubechies 5	0.1325	1.0799	0.0553	12.5711	0.3612
Symlets 5	0.0297	0.2417	0.0106	19.7394	0.6680
Biorthogonal 2.4	0.0287	0.2339	0.0109	19.6235	0.7706

Table 4.6: Reconstruction Quality Comparison of the Full and Half Frequency Spectrum Image Reconstructions.

4.5 Wavelet-based, Parallel Block Multiresolution FBP

In this section, the previously explained wavelet-based approaches, for the fast FBP image reconstruction, are merged into the parallel algorithm that represents the main contribution of this thesis. This algorithm is based in the concept of the accurate reconstruction of reduced-size images, from truncated projection data. From which along different approaches, the ones that involve the application of WT, resulted to be the most appropriate for the fulfilling of this work objectives.

Differently to the previously mentioned approaches, the objective of using the WT in the development of the algorithm presented in this thesis, is not only to achieve accurate reconstructions from truncated data, but also to exploit the parallel multiresolution framework that the two-dimensional FWT is able to provide.

The implementation of the algorithm proposed in this thesis, resembles the approaches proposed by [27, 33, 34]. Through Olson-DeStefano's work in [27], it was possible to understand the non-locality of the FBP, for which the utilization of the WT was first proposed in terms of a numerical algorithm. Delaney and Bresler later proposed a generalised version in [33], where the application of the WT was substituted by the two-dimensional version. Delaney-Berler's approach was crucial in the understanding of the multiresolution representation of wavelet-based Tomography reconstruction. As well as Olson-DeStefano's approach, their work relied in the local properties of the WT.

Farrokh Rashid-Farrokhi, *et al.*, proposed a slightly different multiresolution framework for local Tomography reconstruction, that relied as well in the local properties of the WT [34]. The novelty of such approach was that, for the

reconstruction of ROIs, truncated data was used for both low and high resolution components of projection data.
The three different mentioned approaches, were developed by having the same objective, which consisted in reducing the radiation dose exposure to patients in medical applications. Therefore, they were in the pursuit of achieving accurate resolution reconstructed images, from limited projection data. For such reason, the amount of projection data that could be discarded, led the related research at that time.
After having experienced with the principles of the just mentioned approaches, it was found that for the algorithm proposed in this thesis, truncating data for both high and low resolution components, did not have an important positive impact in the speed enhancement, but negative in reconstruction resolution quality. For this reason we opted for a multiresolution framework in which projection data is decomposed up to a desired scale, being the lower one, preserved as it is, with global measurements.

4.5.1 Projection Data Truncation

Truncation of projection data is performed with the aim of simplifying the reconstruction problem. By splitting projection data into smaller sets of measured values, it is intended to reduce the overall reconstruction time and complexity, through a parallel image reconstruction implementation. Along this description, reduced area images are interchangeably denoted as ROIs or blocks. For the decomposition of Tomography projection data, into a set of n constituent block areas, a simple methodology was developed. Such methodology consists in generating synthetic phantom images, that represent an specific ROI within the acquisition FOV. The RT of the generated image phantom is then calculated, so the area within the Radon space, that the line integrals belonging to the specified ROI engage, is obtained. Such area is the support of the ROI within the Fourier space, which is handled as an image and converted to binary pixel values. The resulting image is used as a template to extract the relevant line integrals, needed for the reconstruction of the ROI; more specifically, an image block. The whole procedure is exemplified in Figure 4.42.

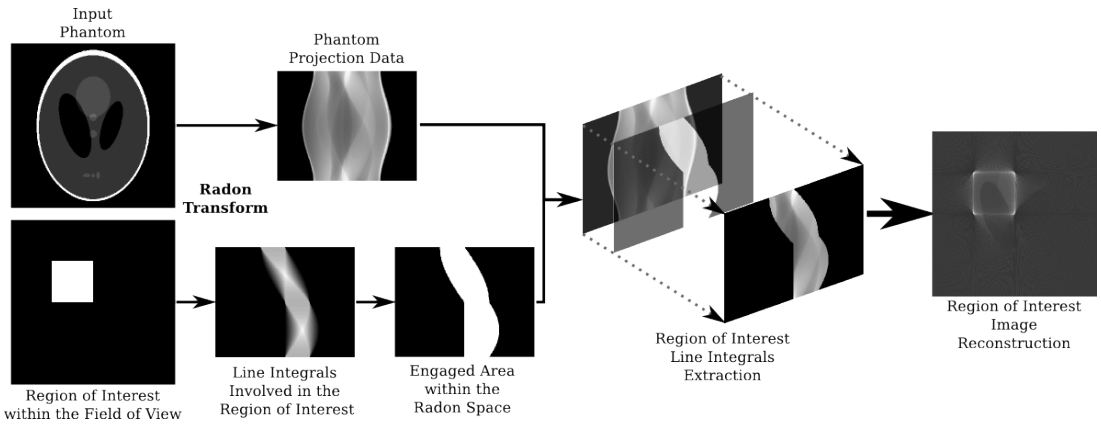
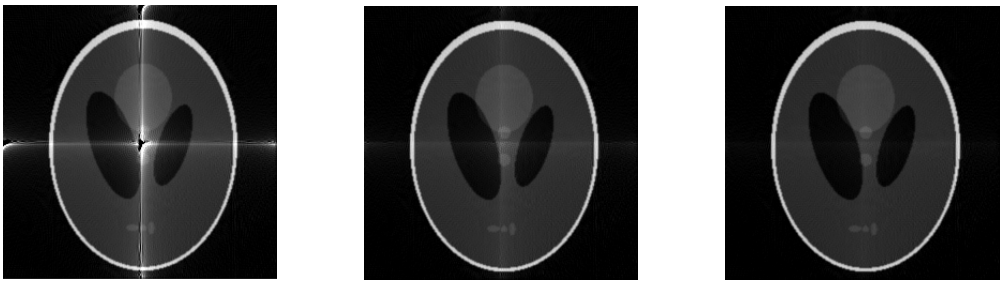


Figure 4.42: ROI Projection Data Truncation.

This truncation approach has been developed with the purpose of avoiding the difficulties involved in the extraction of off-centred ROIs related projection data. In contrast to the methodology briefly mentioned in [27], this alternative method does not need to zero-pad the projection data set, and therefore avoids to handle unnecessary extra data.

It is important to remark that the truncation method represents only an alternative to extract desired data from the projection data set and that it is implicit that truncating a projection data set will lead it to ill-posedness. Reconstructing a ROI by considering only line integrals passing over it, is not well defined, so no matter if data is processed by using the WT, it will always be necessary to gather extra vicinity data [2, 27]. In the example shown in Figure 4.43 the Shepp Logan phantom is decomposed into four constituent blocks, where 4.43a does not gather any extra data from the vicinity, 4.43b gathers 10%, and 4.43c gathers 20%.



(a) No Extra Data.

(b) 10% of Extra Data from the Vicinity.

(c) 20% of Extra Data from the Vicinity.

Figure 4.43: Reconstruction from Truncated Projection Data.

As Figure 4.43 shows, by the gathering of data from the vicinity, it is possible to alleviate the undesired effects of data truncation, which can be clearly perceived only by looking at the figure, without the need of any error metric. Applied to the truncation methodology previously explained, a simple solution was to extrapolate the block image within the synthetic phantom, so an specific extra percentage of data could be gathered, beyond the desired ROI area.

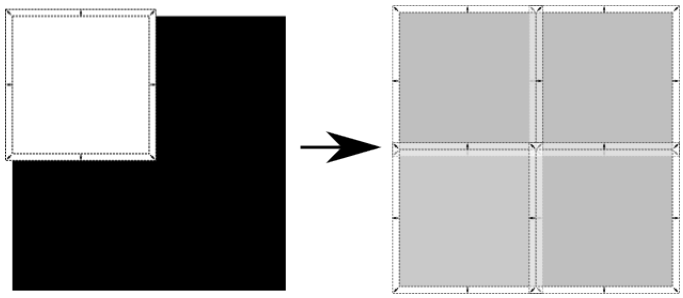


Figure 4.44: Gathering of Extra Data Within the Vicinity.

In terms of the reconstruction procedure, gathered extra data does not represent a considerable increase in the algorithm complexity, as although it increases the amount of data to be processed, the backprojection component stills being dedicated only to the area of the ROI.
In MATLAB, generating the projection data for every block, or conversely, decomposition the projection data set into a collection of block sinograms, was implemented in the following form for a 3-scale multiresolution example.

```
%=================================================================
%---REGIONS OF INTEREST-------------------------------------
%=================================================================
for i = 1:16    %block sinograms
    filename = sprintf('templates/template_1_%d.png',i);
    grid_template{i} = (imread(filename));
end
for i = 1:4     %block sinograms
    filename = sprintf('templates/template_2_%d.png',i);
    grid_template_2{i} = (imread(filename));
end
for j = 1:length(grid_template) %data truncation for l=1
    for i=1:size(sngrm,2)
        sngrm_roi{j}(i,:) = sngrm(:,i).*grid_template{j}(:,i);
    end
```

```
16 end
17 for j = 1:length(grid_template_2) %data truncation for l=2
18     for i=1:size(sngrm,2)
19         sngrm_roi_2{j}(i,:) = ...
               sngrm(:,i).*grid_template_2{j}(:,i);
20     end
21 end
22 sngrm_roi_3 = sngrm; %lower scale, not truncated
```

Where a collection of block phantoms is stored in memory, from which the RT is calculated to create the templates that are used to extract the corresponding data from the input phantom sinogram.

An important parameter in the decision of the number of blocks in which the projection data set is decomposed, depends upon the number of scales. In this algorithm, such information determines the area of the ROIs. According to what was stated at the beginning of this chapter, the lower resolution components are not truncated, so they determine the smaller size of ROIs constituting the higher resolution components.

By keeping consistency with the examples shown along this chapter, a 256 by 256-pixel image is used to exemplify the block distribution. Such image is again decomposed into three scales $l = 1$, $l = 2$ and $l = 3$. The expected size of the approximations and details images, at scale $l = 3$, is of one eight the full resolution image size (i.e. 32x32 pixels), so it is desired that ROIs, at scales $l = 1$ and $l = 2$, be of that size. Scale $l = 2$ details coefficient images, which from the $\downarrow 4$ subsampling involved in the multiresolution decomposition, are expected to be of a size equal to 64 by 64 pixels. Projection data to generate scale $l = 2$ images, will therefore truncated to accomplish four 32 by 32-pixel block images. By applying the same criteria, scale $l = 1$ details coefficient images that are expected to be 128 by 128-pixel size, will be obtained from projection data truncated to sixteen 32 by 32-pixel ROIs. Figure 4.45, illustrates the distribution of the block images, at each scale.

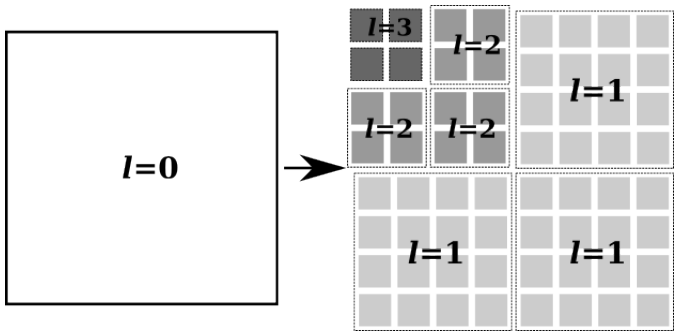


Figure 4.45: Block Distribution of Multiresolution Coefficient Images.

4.5.2 Image Reconstruction from the Wavelet-based, Parallel Block Multiresolution FBP

The final stage in the development of the proposed algorithm is presented in this subsection, which previously proved to achieve fast accurate reconstructions from the inclusion of the two-dimensional FWT. Subsequently from the addition of the parallel multiresolution decomposition features, it was possible to split the reconstruction problem into even less complex processes. As final improvement, it was appropriate to have experienced in exploiting the WT capabilities to perform local reconstruction, so the algorithm was generalised to the parallel block scheme.

The main differences of this parallel block algorithm, with the latter explained reconstruction implementation (i.e. wavelet-based, parallel multiresolution FBP), rely in the handling of the smaller block images, which at the beginning are obtained through the truncation of projection data.

In order to achieve a parallel description of the algorithm, in contrast to the previous reconstruction implementations, the sequential MATLAB code was split into elements, chosen among those that do not rely on values obtained during runtime, and those that do not change during the execution of the algorithm, so can be calculated a priori. Figure 4.46 has been included to illustrate the parallelism that can be achieved through the implementation of this algorithm.

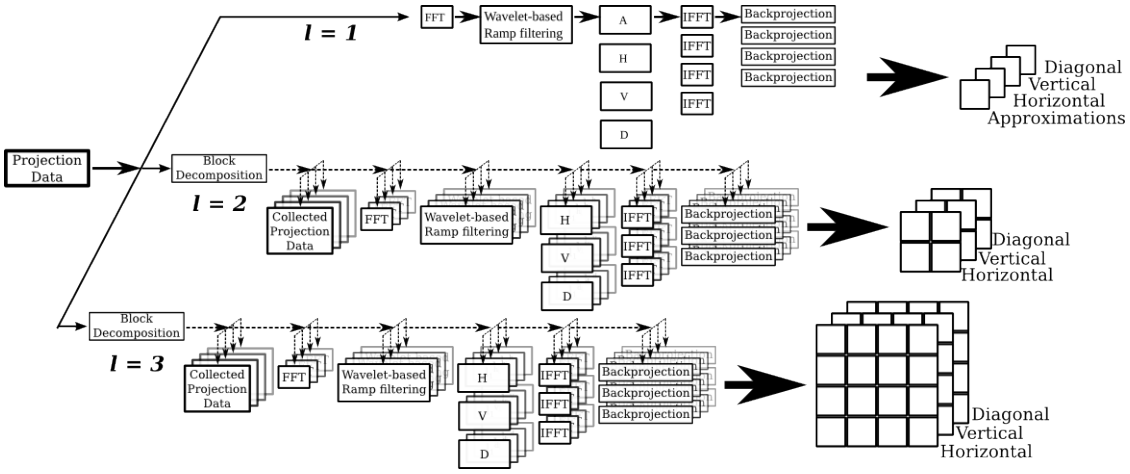


Figure 4.46: Wavelet-based, Parallel Block Multiresolution FBP Algorithm.

Another feature that vary this implementation from the previous ones, is the way in which backprojected block images are merged in the end, to fill the pixel grid that corresponds to every details coefficient images, at a specific scale. In MATLAB such operations did not represent any additional computation cost, given its nature capabilities to handle data arrays, so block images were easily tiled to form the corresponding scale details coefficient images.

```
%=========================================================================
%---REGIONS OF INTEREST TILING-------------------------------
%=========================================================================
for j = 1 : (size(imagend,1)/4) : size(imagend,1)
  c = 1;
  for i = 1 : (size(imagend,1)/4) : size(imagend,1)
   imagend((j:cc*size(imagend,1)/4),(i:c*size(imagend,1)/4)) ...
       = imgd_nrm{n};
   imagenv((j:cc*size(imagenv,1)/4),(i:c*size(imagenv,1)/4)) ...
       = imgv_nrm{n};
   imagenh((j:cc*size(imagenh,1)/4),(i:c*size(imagenh,1)/4)) ...
       = imgh_nrm{n};
   c = c+1;
   n = n+1;
  end
  cc = cc + 1;
end
```

The full MATLAB code of the Wavelet-based, Parallel Block Multiresolution

FBP Algorithm, has been included in Appendix A.5.
In order to preserve the format shown in the explanation of the previous reconstruction implementations, an example of the parallel block algorithm reconstruction has been included to conclude this section. The next Figure shows the scale $l = 3$ approximations and details, as well as the tiled details images for scales $l = 2$ and $l = 3$. Scale $l = 1$ projection data was truncated to sixteen blocks images, which were obtained from truncated data using block phantoms with 30% extra data. Scale $l = 2$ projection data was truncated to four blocks images, calculated from 20% extra data, block phantoms.

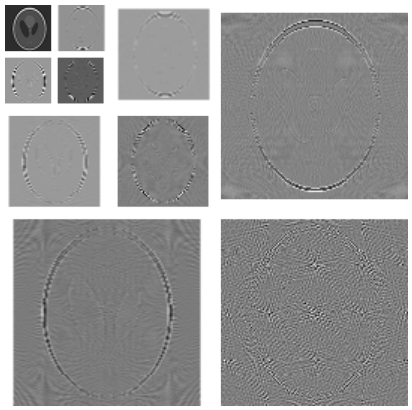


Figure 4.47: Tiled Block Reconstructed Multiresolution Coefficient Images.

The backprojection time of each of the block images showed to be within a very narrow threshold, just as it was expected, as all of them have the same area. The time measurements were performed as in the previous cases, once again taking into account the time spent into the two-dimensional FWT synthesis. For scales $l = 1$ and $l = 2$, the displayed time equals to the processing of a single block, so in order to achieve such time saving in the overall reconstruction, it is once again important to mention that a parallel implementation is required.

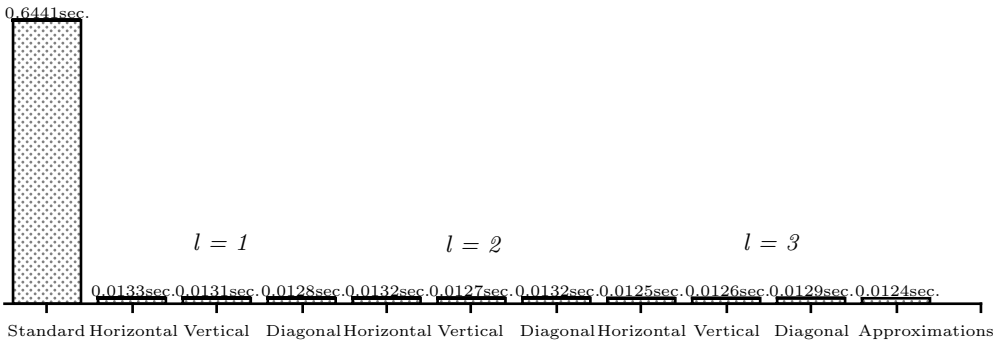
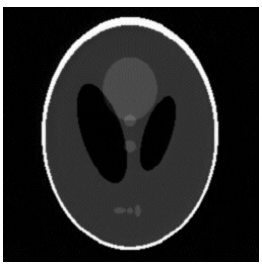


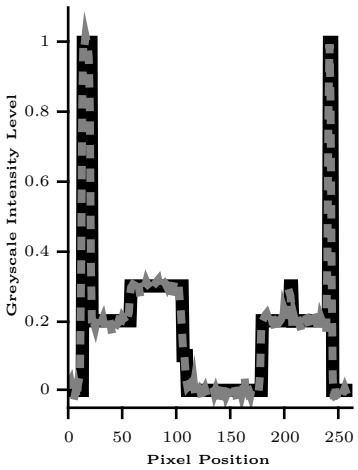
Figure 4.48: Wavelet-based, Parallel Block Multiresolution Reconstruction Time Comparison.

The synthesis of the full size and full resolution image does not suffer any considerable changes, compared to the parallel multiresolution approach. The inverse FWT procedure is performed exactly the same way as it is performed for the parallel block scheme modification. The quality results, as well as the time results, are satisfactory. It can be seen in Table 4.7 that error is slightly increased compared to the previous approaches (wavelet-based and multiresolution), nonetheless reconstruction speed is dramatically improved by achieving a full size and full resolution image reconstruction in around 48 times faster than the standard FBP, once again by assuming a parallel computation of the FWT coefficient images. In terms of quality, even after several manipulations to reduce to computational load of the reconstruction method, it was possible to recover an accurate estimation of the Shepp Logan input phantom.

(a) Input Phantom.



(b) Parallel Block Reconstructed Image.



(c) Greyscale Intensity along $y = 155$.

Figure 4.49: Comparison Between the Input Phantom and the Parallel Block Reconstructed Image.

Considered Frequency Spectrum	AVERR	NABS	MSE	PSN	SSIM
Standard	0.0174	0.1419	0.0018	27.3788	0.8663
Parallel Block	0.0302	0.2463	0.0110	19.5928	0.7372

Table 4.7: Reconstruction Quality Comparison of the Full and Half Frequency Spectrum Image Reconstructions.

4.6 Summary

In this chapter the FST was applied and the inverse RT in the form of the FBP reconstruction method, along with the two-dimnsional FWT, to achieve decomposition of the measured projection data. While it is not trivial to identify the FBP reconstructions as physical objects (i.e. an object density function), they can be interpreted as a set of characteristics of a single object, which together manifest the spatial details of the complete physical object. Thus the spatial characteristics of the said physical object are obtained by an inverse two-diemnsional FWT. It is worth noting that while the sampling of the RT is determined by the measurements, the sampling of the FWT is not experimentally dependent and therefore can be chosen in such a way that it does not compromise the image resolution defined by the RT framework.

The main motivation in applying the backprojection operator, not on the projection but on the FWT transformed data, is to avoid the support spread of projection data that is caused by the standard ramp filtering. Thus the filtering stage of the standard FBP is extended by means of compactly supported wavelet and scaling functions, presenting a certain amount of zero moments to avoid the support spread of projection data after filtering.
The involvement of two-dimensional FWT into the FBP framework, is acceptable in terms of affecting the accuracy of reconstruction. The gain in speed, however, is of around 48 times faster, by assuming a parallel implementation.

Chapter 5

Hardware Implementation of the Wavelet-Based, Multiresolution Parallel Block Reconstruction Algorithm

Either in Electronic or Computational Engineering, it is natural within the design process of a wide range applications, that have the intention of being implemented in real practice, to face the dilemma of what is the most appropriate computing platform to be used. Such decision may depend upon some variables that range from the affordability to employ a determined technology, to the accomplishment of the application requirements. Through a broad classification in terms of flexibility and performance, computer architectures can be categorised in to three different classes: GPAs, DSAs and ASAs [50, pp. 1-12].

A GPA is a fixed structure formed by three main elements: memory, arithmetic-logic unit and control unit. The idea is that, through a properly programmed control unit, any kind of operation may be computed without the need to modify hardware elements. In GPAs, operations are performed from instructions contained in a coded program that is loaded into memory. The execution of every instruction is usually performed in five sequential steps, that constitute an instruction cycle. The five steps commonly consist in: the instruction reading from memory, the decoding of the read instruction, the reading of needed operands to execute the instruction, the instruction execution

with the read operands, and the storing of the obtained result into memory. In GPAs, the execution of the coded program is performed in a sequential fashion, instruction after instruction. GPAs are commonly built based on The Von Neumann and Harvard architecture models [143]. The clearest example of a GPA is the CPU.

A DSA is a kind of architecture based on the general purpose model, but with elements adapted for a certain class of algorithms. In DSAs, the arithmetic-logic unit is designed to execute a common set of operations exclusive of the algorithms in the given class. The most common example of a DSA is the Digital Signal Processor (DSP). DSPs are particularly characterised for having an arithmetic-logic unit designed to perform floating-point operations, as well as one or more multiply-accumulates in a single execution cycle [144]. Because DSAs are based in the general purpose model, instructions are as well sequentially executed.

ASAs are completely different from the previously mentioned approaches. ASAs are designed for a defined application, and are not dependent of instruction cycles, instead the instruction set is directly implemented in hardware in terms of functional units, specified by the application. ASAs design belong to the so-called *Spatial Computing*, in which the necessary functional units for the application computation are available in the surface of the final device. Because ASAs are used for dedicated implementations of a specific algorithm, they can provide higher improvements in performance. Although, high performance is obtained at the expense of flexibility, in as much as its design can not be modified after being manufactured. ASAs are widely encountered within embedded systems, as they are mainly employed in a narrow domain of applications. ASAs are usually implemented as a single chip device: the ASIC [50, p. 6], [145].

From the previous categorisation of computer system architectures, it is possible to identify their importance by means of flexibility and performance. GPAs and ASAs lie at extreme sides.

Flexibility is dominated by GPAs, because in these kind of architectures, the application is adapted to the hardware, in order to be implemented. DSAs share most of the flexibility capabilities, with the difference that their application field is more specific. Although the lack of performance in GPAs emerge from the impossibility to execute instructions in parallel, as well as from

the disadvantage of instruction cycles in the computation of huge sets of data. On the opposite side lies ASAs, which high performance is feasible because, in this kind of architecures, the hardware is designed in terms of the specified application. ASAs flexibility is very low or even null, because they are designed to fulfil only a determined application, and the design cannot be modified after being manufactured [145].
For a wide class of applications, it would be ideal to have a device sharing both the flexibility of the GPAs, as well as the performance of the ASAs. A device capable to adapt to the application in terms of its hardware structure, and to modify all or part of such structure at compile-time or at run-time.

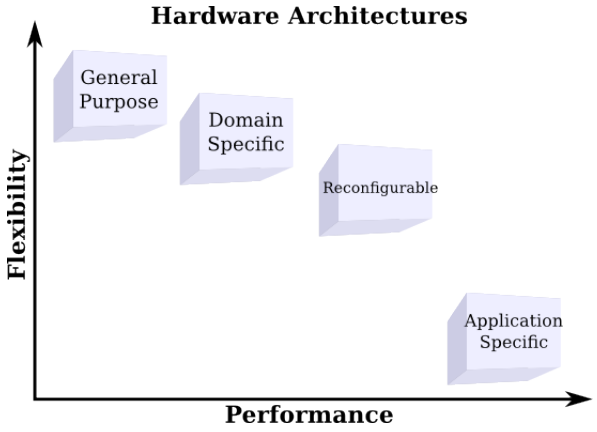


Figure 5.1: Flexibility vs. Performance.

FPGAs are reconfigurable devices that allow the design of functional units that can be executed in parallel. FPGAs not only provide high performance in an application execution, but also the flexibility to either modify a design during development, or upgrade an existing implementation to better match with the application requirements. Nowadays FPGAs represent an alternative to ASICs and GPA processors [64].

5.1 The Utilisation of Computer Architectures in Tomography

Since the early years of Tomography development, ASICs have been developed by Tomography companies. Although the production of ASICs is efficient, it is also too specialised and can only be affordable by companies that target

sufficiently large markets [51, 52].
GPAs, or more specifically Personal Computers (PCs), are commonly used by researchers in the pursuit to experience on new algorithms and/or acquisition geometries, as well as a preamble to a further specialised implementation. A PC cluster is a more scalable approach to experience on parallel processing, it consists of multiple GPAs connected by a network to work together. PC clusters offer high performance at low cost, although they are limited by communication speed [41].
An extension of GPAs platforms are the multicore devices that consists either on multicore processors or GPUs. GPUs are devices created under the stream processing paradigm [54, 57]. Stream processors are the approach created to fill the gap between flexible GPA solutions and high performance ASA approaches. Some examples of Tomography implementations in DSAs are in [54, 59–63].
For RH, although the FPGAs can not achieve the performance of ASAs, they have been proven to be a feasible option in the parallel implementation of large volume tomographic data image reconstruction applications [67, 68, 71].

5.2 FPGA Implementation of the Wavelet-based, Muliresolution Parallel Block Reconstruction Algorithm

In an ideal scenario, in order to achieve a hardware implementation, that would exactly accomplish the requirements of the high-level algorithm design, it would be necessary to employ a hardware design platform that may allow the creation of components with a high degree of customisation. As previously mentioned, the computer platform that may perform better in the creation specific hardware implementations are ASAs, or more specifically, ASICs. Although for an ASIC-based implementation, it is not only required to have a high degree of knowledge, but also specialised Computer Aided Design (CAD) software and production equipment. Such features make ASIC production affordable only for a narrow field of applications, like big consumer markets, or government-funded projects.
The closest alternative that follows ASICs, in terms of customisation, are the FPGAs. Even though FPGAs can not achieve the full custom capability of the

ASICs, their design cycle is simpler and involve far-less expenses; moreover, FPGAs are reconfigurable, making them the proper alternative for testing and prototyping [65].
In order to test with a true parallel implementation of the designed algorithm, an FPGA-based system approach, has been developed in resemblance with the designed MATLAB reconstruction algorithm. The objective of such implementation, is to overcome the MATLAB inherent sequential limitation, that was present in the high-level design of our reconstruction algorithm.
For this system, the FPGA implementation of the full algorithm has been limited to what represents the smaller processing unit within the parallel scheme, the processing of an angular projection. The main purpose is therefore to carry out the design involved in the essential elements that compose a unit that can be replicated up to the requirements of a certain application, like: the characteristics of the input projection data set (i.e. number of angular projections and the number of line integral per projection), and the desired multiresolution depth, which also indicates the number of parallel block images in which the full-resolution reconstruction procedure will be decomposed.

5.2.1 Preliminary Design Specifications

Similarly to the high-level design in MATLAB, the system design was split into three main components: the filtering of projection data by the multiresolution wavelet-modified ramp filters, the backprojection of filtered data, and the inverse two-dimensional parallel FWT of coefficient images.
To make the the hardware implementation system as flexible as possible, it was designed through a modular scheme system, where VHSIC Hardware Description Language (VHDL) was used as the hardware description language. For arithmetic operations, the fixed point representation was used, in order to save resources and consequently latency.
For data values that do not depend on calculations during runtime (e.g. filter coefficients), they have been previously calculated and translated to the fixed-point representation, by using MATLAB. The decimal to fixed-point and vice-versa operations, were performed through the "two's complement binary strings" set of functions, developed by Drew Compson, and available in MATLAB Central [146].
The VHDL system was designed by using the Xilinx Integrated Synthesis

Environment (ISE) Design Suite v13.4, for which the available of-the-shelf development board, the Virtex-6 ML605, has license [147]. To validate results and behaviour of the system components, the ISE integrated ISim simulator, was used. The ISE CORE Generator IP, was employed for the instantiation of pre-designed Semiconductor Intellectual Property Core (IP) modules, and the Memory Generator to create .coe data files, that considerably ease the pre-synthesis data loading of Read-only Memorys (ROMs) [148].

5.2.2 Wavelet-based Filtering of Projection Data, VHDL System

The filtering of projection data component of the VHDL system, was formulated by having the elements shown Figure 5.2 below.

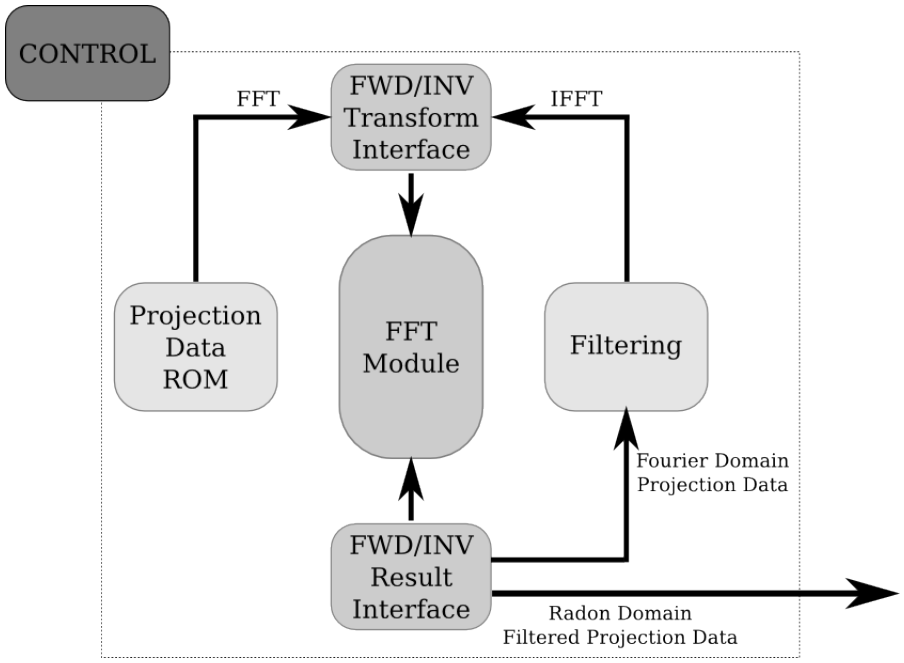


Figure 5.2: Projection Data Filtering System.

Where projection data to be processed, is stored in a ROM that feeds an IP-instantiated, FFT module. Such module is able to perform both forward and inverse FFT, therefore it is used interchangeably for both operations. The request to either perform a forward or inverse FFT operation is managed by two interfaces located at the input and output of the module. The filtering component is formed by an IP-generated complex multiplier module, and two

ROMs that store the real and complex wavelet-modified, ramp filter coefficients. The execution sequence of all the modules is managed by a state machine, that comprises the control module. The output of the system is the filtered angular projection in Radon domain, and the input to the Backprojection system.

5.2.2.1 Storage and Loading of Angular Projection Data

This component serves to store the angular projection data required to be filtered by the system. Filtering is performed in the Fourier domain, so this component is used to send projection data values to the FFT module. In addition to the ROM, the component comprises an interface entity that manages the data requests that are received from the FFT module. The RTL elements of the projection data storage components is shown below in Figure 5.3.

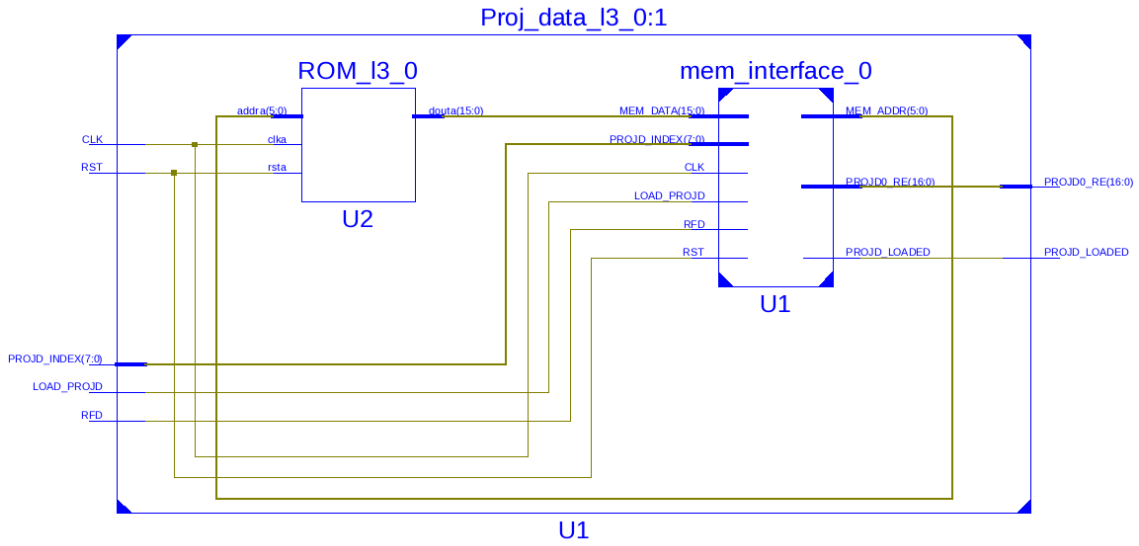


Figure 5.3: Angular Projection Component.

The ROM entity is a single port memory instantiated by the CORE generator, and filled only with non-zero values of an angular projection through a .coe file generated by the Memory Generator. Angular projection values are generated by the RT of the Shepp Logan phantom, in MATLAB. By non-zero values, it means that in order to reduce storage resources, only values within the support of the sinogram, are considered as shown in the Figure 5.4.

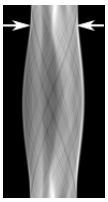


Figure 5.4: Non-zero Values.

Figure 5.5 show a plot of projection data at angle $\theta = 8$ obtained from the RT of the 64 by 64 Shepp Logan phantom, that was employed as example to verify the correspondence between the MATLAB algorithm and the VHDL system implementation. Within the plot, MATLAB values are represented through the dotted line, while the non-zero values that were loaded in the instantiated ROM are represented through a star symbol.

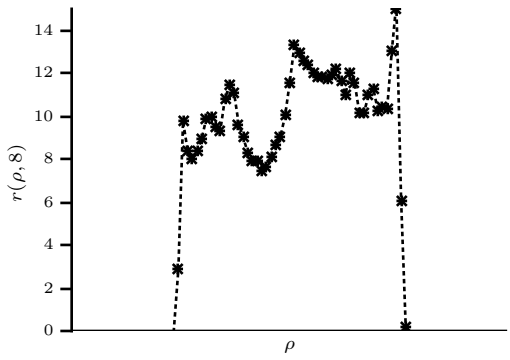


Figure 5.5: Projection Data $r(\rho, 8)$.

The description of each of the module's inputs and outputs is given in Table 5.1.

Port Name	Direction	Description
CLK	Input	Clock
RST	Input	Reset
PROJD_INDEX	Input	Projection data memory index
LOAD_PROJD	Input	Load projection data Enable
RFD	Input	Ready for data Indicator
PROJD0_RE	Output	Projection data requested value
PROJD_LOADED	Output	Projection data loaded

Table 5.1: Angular Projection Module Ports Description.

As the index of the angular projection values is important to feed the FFT

module, no matter if they are zero or not, information about the support width of the sinogram is introduced in the memory interface, so the index can be appropriately managed. Moreover, the memory interface also manages the zero-padding needed by the FFT. The reset signal *RST*, as well as the read enable *LOAD_PROJD*, are managed by the state machine in the Control module.

5.2.2.2 FFT Module

The Xilinx LogicCORE FFT module is based on the Cooley-Tukey algorithm, it can perform both the forward and the inverse FFT. This module is customisable to different parameters like transform sizes, and data sample and phase factor precision. Different arithmetic representations are also available. The module can also be customised either to maximise the transform speed performance, or under minimum resources utilisation.
This module allows the processing of twelve independent transform calculations, a feature that can increase the parallel capabilities of the application with less resources utilisation. Another important feature of the module is that the latency between data transfers, is minimised by the fact that the FFT module employs on-chip memory (i.e. block Random-Access Memory (RAM) or distributed RAM). For more information about the capabilities and configuration of the FFT IP core, the reader can refer to [149].
The FFT module considerable eased the design of the proposed system, not only because it was already available for implementation, but also because it is designed to maximise the resources utilisation of the FPGA device, and to deliver accurate results. To meet the minimum resource utilisation requirements of the proposed system, the FFT was instantiated with the configuration parameters shown in Table 5.2. Where the input data length is designed for a 256-elements angular projection sequence, obtained from a 64 by 64 Shepp Logan phantom. In contrast to the examples shown in Chapter 4 for the high-level computer algorithm, in the hardware system we opted for a reduced size input phantom. Such decision was made with the objective of simplifying the design process. Nonetheless, all the parameters that derive from the input data sizes, can be easily reconfigured before the VHDL synthesis.

Parameter	Assigned Value
Transform Length	256 samples
Target Clock Frequency	200 MHz
Implementation	Radix-2 Lite for minimum resources
Data Format	Fixed-Point
Input Data Width	17 bits
Input Data Timing	3 clock cycle offset
Data Memory	Block RAM
Phase Factors Memory	Block RAM

Table 5.2: FFT Module Configuration Parameters.

Figure 5.6 shows the FFT entity that results from the instantiation performed, by considering the configuration parameters of Table 5.2.

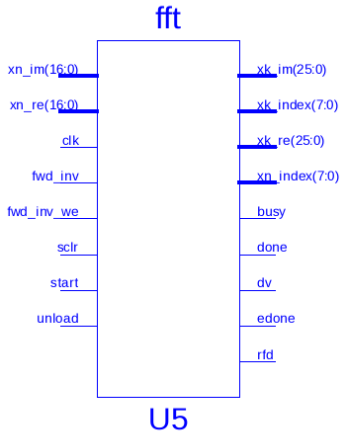


Figure 5.6: FFT Module.

The instantiated FFT module of Figure 5.6, has been configured to process a single input data angle, which can be either complex or real. In the complex case, real values are given at port *xn_re* and imaginaries at *xn_im*. The description of every port is given in Table 5.3.

Port Name	Direction	Description
xn_im	Input	Input Data Bus (Imaginary)
xn_re	Input	Input Data Bus (Real)
fwd_inv	Input	Control for Fwd or Inv Transform
fwd_inv_we	Input	Write enable for fwd_inv
sclr	Input	Synchronous Reset
start	Input	FFT Start Signal
unload	Input	Result Unloading
xk_im	Output	Output Data Bus (Imaginary)
xk_index	Output	Index of Output Data
xk_re	Output	Output Data Bus (Real)
xn_index	Output	Index of Input Data
busy	Output	Activity Indicator, Transform ongoing
done	Output	FFT Complete Indicator
dv	Output	Data Valid Indicator
edone	Output	Early Done Indicator
rfd	Output	Ready for Data Indicator

Table 5.3: FFT Module Ports Description.

The FFT module consists of four main phases along its operation: the setting-up and initialisation, the input data loading, the data processing and the resulting data unloading. For the setting-up, it is only necessary to provide the module with a high pulse at the *fwd_inv_en* port, combined with the *fwd_inv* indicator, to let the module know which one is the desired transform (low value for forward, and high value for inverse) to be executed. The initialisation is simply performed by a high value given at the start input port.
Once the module has successfully initialised, it will respond with a ready for data *rfd* high value, along with the index of the first requested data element. Both of these events occur at the same clock cycle. This is the event that indicates that the module can continue with the input data loading stage. Figure 5.7 shows the behaviour of the signals involved in both phases setting-up/initialisation and data loading. The sequence is managed by the control module, which through the states *fwd_idle*, *fwd_init*, *fwd_start* and *fwd_loading*, can achieve the proper operation of the FFT module initialisation and data loading. In *fwd_idle*, the system starts with enabled *rst* and *sclr* signals, to avoid any undetermined value during initialisation. In the state *fwd_init*, the signals *fwd_inv_en* and *fwd_inv* are configured. In the state *fwd_start* the FFT module is indicated to start loading data. Finally, in state

fwd_loading the machine enables the angular projection module to start receiving *xn_index* addresses to return stored data elements.

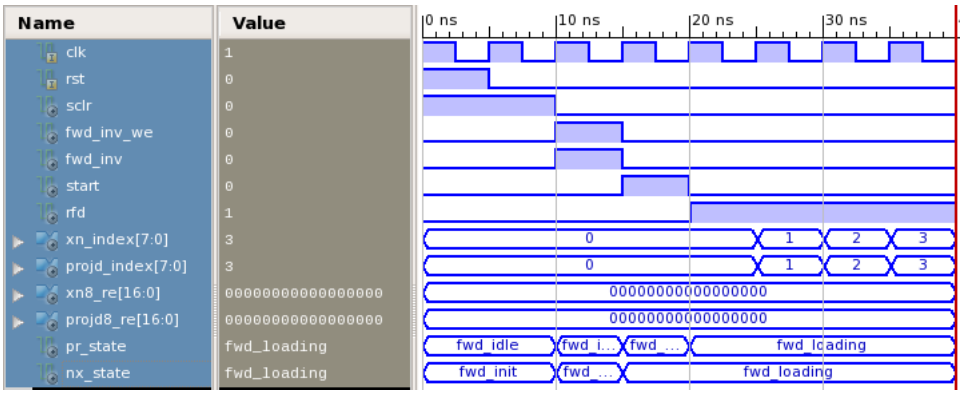


Figure 5.7: FFT Module Initialisation.

As it was previously mentioned, the FFT module was configured to receive data with a 3-clock cycle offset, after the sent *xn_index*. In this example, the angular projection component has stored data, which sequence has its first non-zero value in position 25, therefore after receiving the address *xn_index*= 25, the value will be ready for the FFT module three clock cycles later. This is shown below in Figure 5.8.

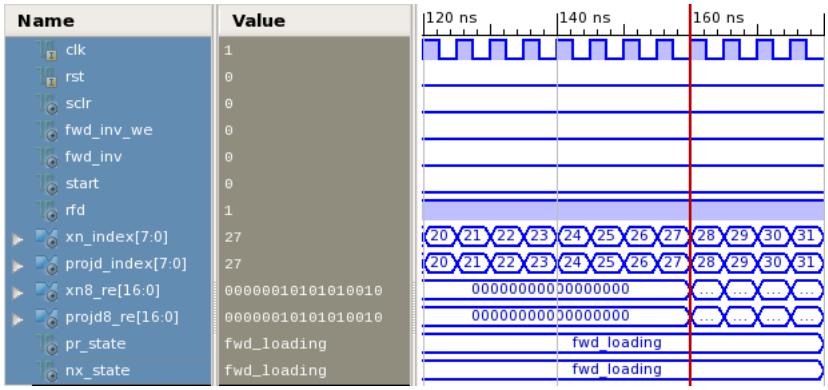


Figure 5.8: FFT Data Loading.

Once that the FFT has reached up to the last input data element, the ready for data indicator *rfd* turns to zero and the *busy* indicator turns to high. Both of them happen at the same clock cycle and indicate that the loading phase has finished, and therefore the data processing has started. Figure 5.9 illustrates such behaviour.

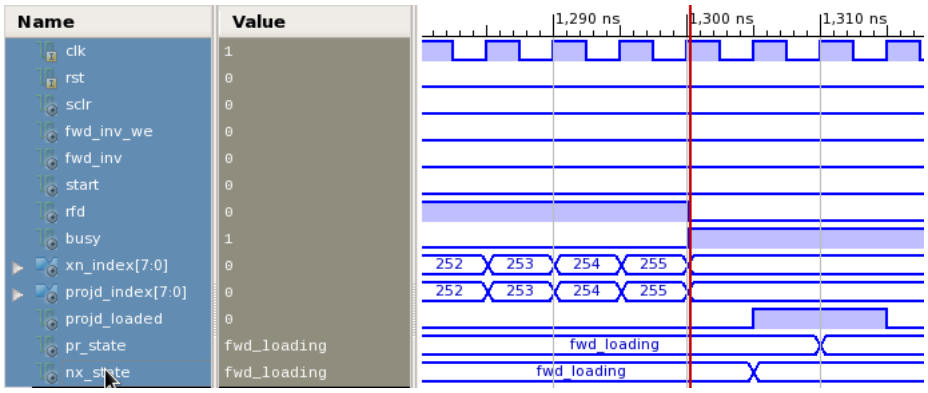


Figure 5.9: FFT End of Data Loading.

The *busy* indicator will remain active up to the end of the data processing stage. After finishing to calculate the transform values the FFT module will turn on the *done* indicator to let the system know that the process has finished, and that is ready to unload the result sequence. This is shown below in Figure 5.10, where the change in the *done* signal is barely visible, at the same time at which *busy* turns to zero.

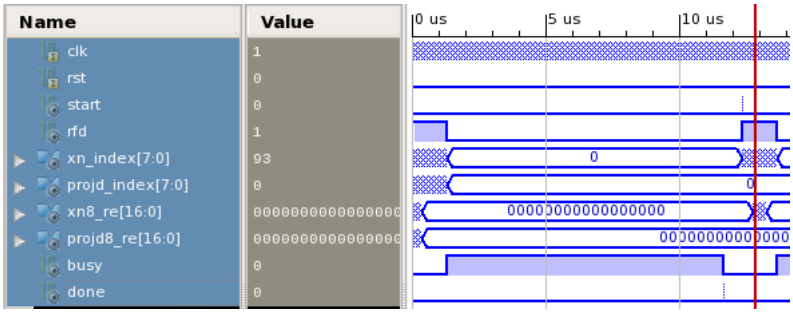


Figure 5.10: FFT Data Processing.

At this point, the control component requests an unload and changes to a state that waits for the activation of the data valid *dv* indicator. The activation of *dv* happens at the same time, at which the FFT module starts sending the output data address *xk_index* and complex values *xk_re* and *xk_im*. Such behaviour is demonstrated in the Figure 5.11.

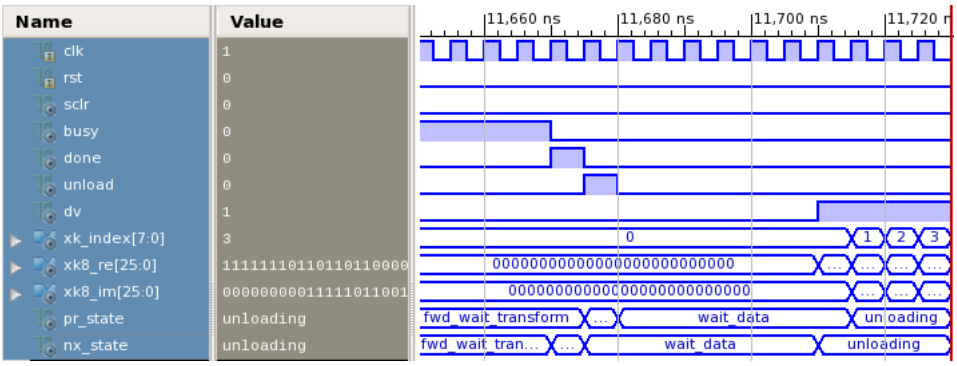
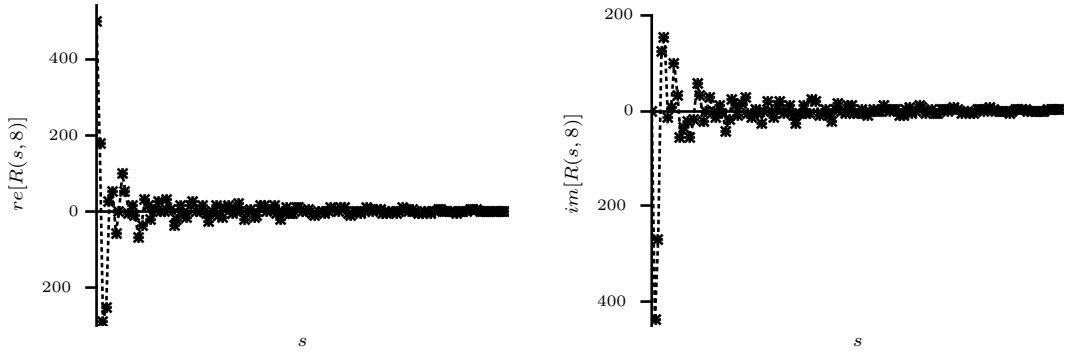


Figure 5.11: FFT Data Unloading.

In the proposed system the FFT module has been implemented to perform both forward and inverse operations. The forward operation is required for the input angular projection data, whilst the inverse transform is required to process the filtered projection data. The finalisation of the unloading phase depends therefore upon the type of transform that has been carried on. If the result corresponds to the forward transform, the unloading will last up to reaching the half+1 value of the resulting sequence, as the filtering will be performed only on the positive side of the frequency spectrum, as previously explained in Chapter 4.

Figure 5.12 shows the correspondence between the FFT of projection data output $R(s, 8)$, obtained from both the MATLAB algorithm and the VHDL ISE ISim simulation. Again, the dotted line represents the MATLAB calculated values, while the star symbol belong to the VHDL implementation.



(a) Projection Data in Fourier Domain $Re[R(s, 8)]$.

(b) Projection Data in Fourier Domain $Im[R(s, 8)]$.

Figure 5.12: FFT of Projection Data Example Result.

On the other side, if the result belongs to an inverse FFT calculation, the

sequence will be truncated up to the original length of the angular projection, before being zero-padded. Figure 5.13 is a plot of the obtained filtered projection data at angle $\theta = 8$ in the Radon domain. By keeping the same format of the example (i.e. dotted line representing the MATLAB calculated values, and the star symbols for the VHDL implementation), it can be noticed that not only the correspondence between both platforms is achieved, but also how the hardware implementation keep only the number of samples specified by the size of the input sinogram.

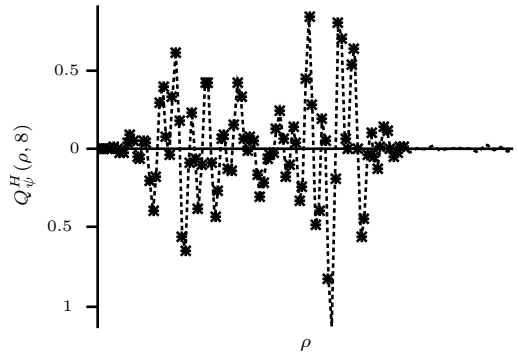
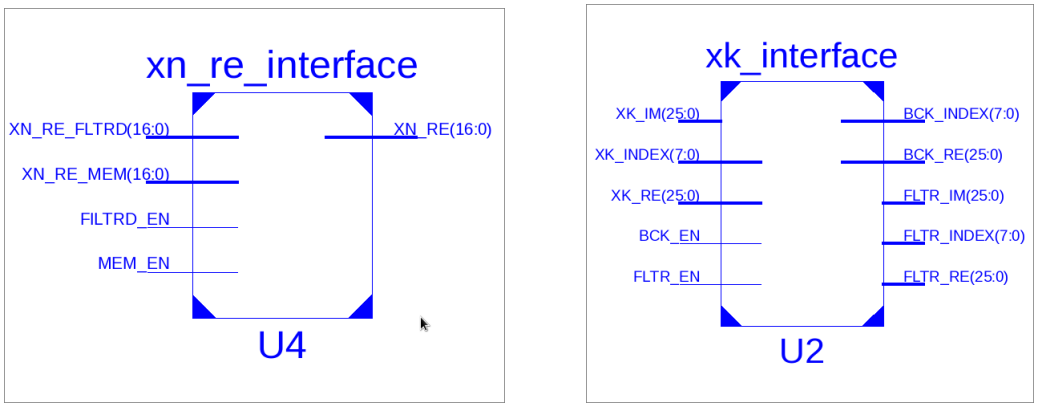


Figure 5.13: Horizontal Filtered Projection Data in Radon Domain $Q_\psi^H(\rho, 8)$.

Either for the forward or inverse FFT, the module follows the same sequence, with the only difference that the input data port will be connected to the angular projection module output, in case of a forward transform. For an inverse transform, the angular projection module output, will be connected to the filtering component output. At the output data port, if the result belongs to a forward transform, it will be connected to the filtering component input. Conversely, if the FFT result belongs to an inverse transform it will be connected to the system output (backprojection system). Such situations are controlled by interfaces situated at the input and output data ports of the FFT module. Figure 5.14 below, shows the RTL blocks of both interfaces.



(a) Input Interface. (b) Output Interface.

Figure 5.14: FFT Module Data Ports' Interfaces.

5.2.2.3 Filtering in Fourier Domain

The filtering component is probably the most complex element within the system, and it comprises four main elements: a complex multiplier, two memories that store the wavelet modified ramp filter coefficients in Fourier domain, and a control entity that manages the indexes of filter coefficients. At the same time, the entity that stores the real filter coefficients is a subsystem that reuses the RAM to store filtered projection data. The RTL diagram of the filtering system is shown below in Figure 5.15.

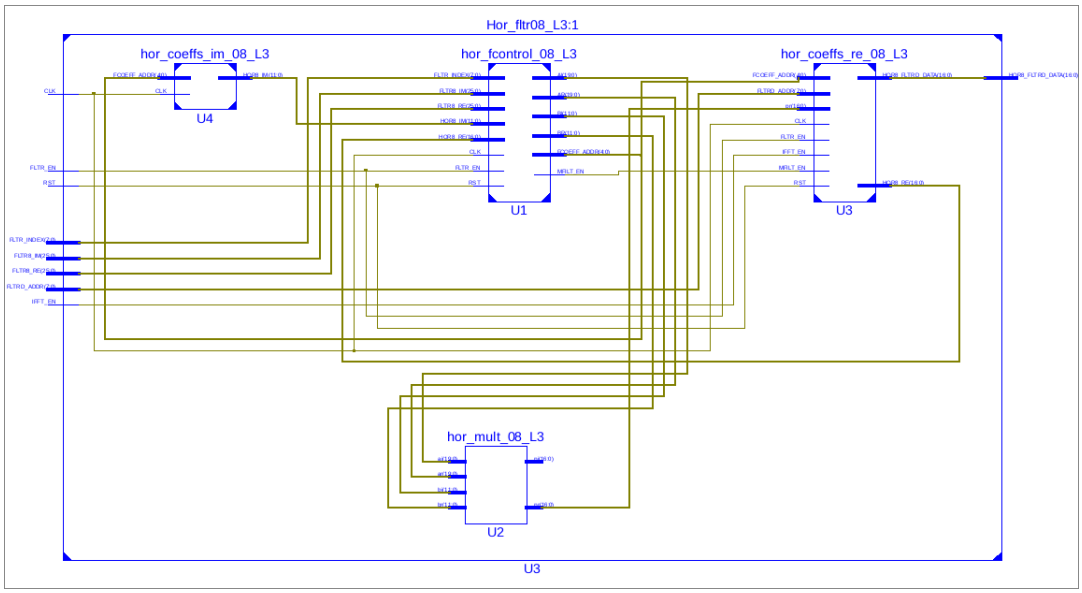


Figure 5.15: Filtering Component system.

Similarly to the storage of angular projections, the stored filter coefficients correspond only to non-zero values. Although for the filter coefficients the manipulation is more complex as the non-zero values are located along sparse intervals. This is shown in Figure 5.16, through the absolute value of the horizontal filter support $|W_{\psi}^{H}(s,\theta)|$.

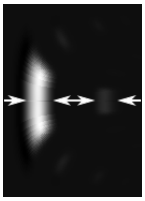
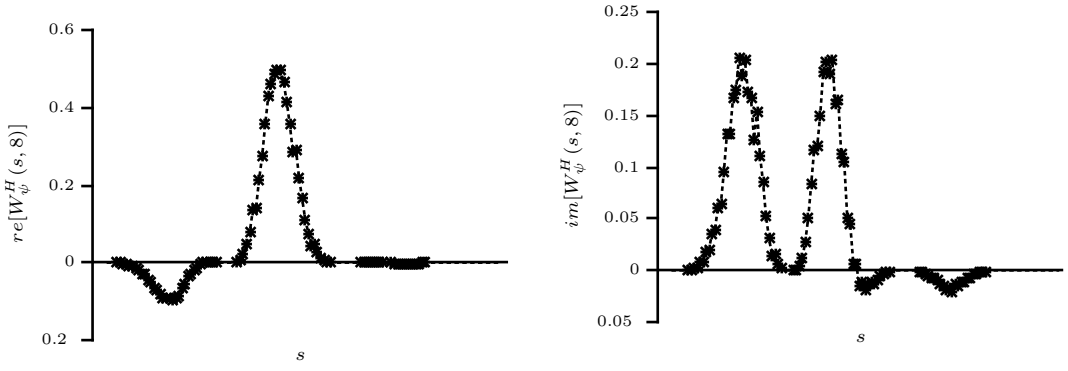


Figure 5.16: Horizontal Filter Support Along Projection Angles.

To manipulate the indexes of the filter coefficients, the control entity analyses the address requests and decides between delivering a value from memory, according to the given interval information, or returning zero values.
The imaginary filter coefficients are stored in a single-port ROM that is instantiated in a similar manner to the angular projection ROM. For the real values a two-port RAM memory is employed to be reused as storage for the resulting filtered values.
The plot in Figure 5.17 show the values that correspond to real and imaginary parts of the horizontal filter $W_{\psi}^{H}(s,8)$. In the plot, the dotted line represents the filter coefficients calculated in MATLAB, while the star symbol represent the non-zero filter coefficient values chosen to be stored in VHDL implementation.



(a) Real Part of Horizontal Filter $\Omega_{\psi}^{H}(s,8)$.

(b) Imaginary Part of Horizontal Filter $\Omega_{\psi}^{H}(s,8)$.

Figure 5.17: Low pass and high pass filters.

In the VHDL implementation, the RAM loaded with the real valued filter coefficients is managed through input/output interfaces. The real filter coefficients RAM-subsystem is shown below in Figure 5.18.

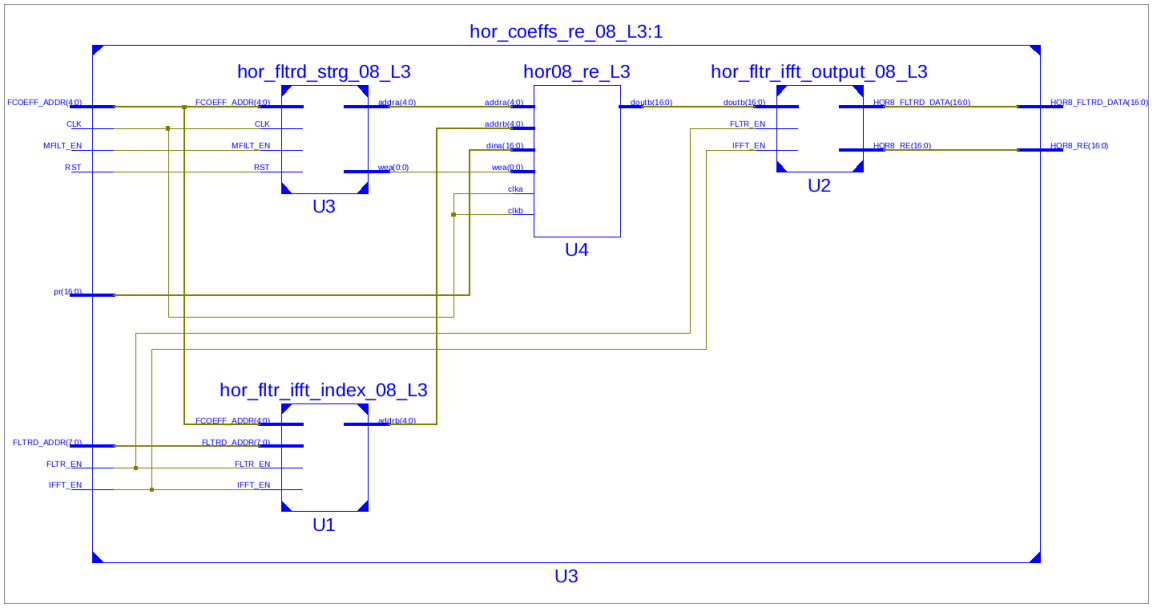


Figure 5.18: Filter Real Value Coefficients Subsystem.

As such subsystems store all the values that result from the angular projection system, it feeds the required inverse FFT to calculate the Radon domain filtered angular projection. To feed the FFT module, the system manipulates the order of the stored filtered projection values to its complex conjugate reverse-order, as the filtering is only performed on the positive side of the frequency spectrum. Another important component is the complex multiplier, which is also an instantiated LogiCORE IP. This complex multiplier can be configured in terms of the arithmetic representation, as well as the widths of the operands and the result output. Among several configuration options, the complex multiplier allows to configure the latency between the multiplication and the result output. For the implementation in this system, the complex multiplier (Figure 5.19) was chosen to be implemented by using Look-up Tables (LUTs), and with a latency of zero clock cycles. More detailed information about the complex multiplier can be found in [150].

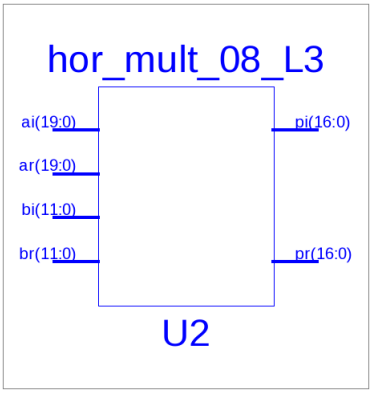


Figure 5.19: Complex Multiplier.

The port description for the complex multiplier, is given in Table 5.4.

Port Name	Direction	Description
ar	Input	Real Operand A
ai	Input	Imaginary Operand A
br	Input	Real Operand B
bi	Input	Imaginary Operand B
pr	Output	Real Product
pi	Output	Imaginary Product

Table 5.4: Complex Multiplier Ports Description.

The port description for the filtering component system, is given in Table 5.5.

Port Name	Direction	Description
FLTRD_ADDR	Input	Index for Filtered Data Requests
FLTR_INDEX	Input	Index for Filtering
FLTR_IM	Input	Imaginary Data for Filtering
FLTR_RE	Input	Real Data for Filtering
FLTR_EN	Input	Filtering Enable
IFFT_EN	Input	Release Result for IFFT Enable
FLTRD_DATA	Output	Filtered Data Output

Table 5.5: Filtering Component Ports Description.

The filtering component starts its operation when the *unload* indicator for the FFT module is activated. At the same time, the filtering is enabled through the activation of the *FLTR_EN* enable port. Figure 5.20 below, show the signal sequence of such an operation.

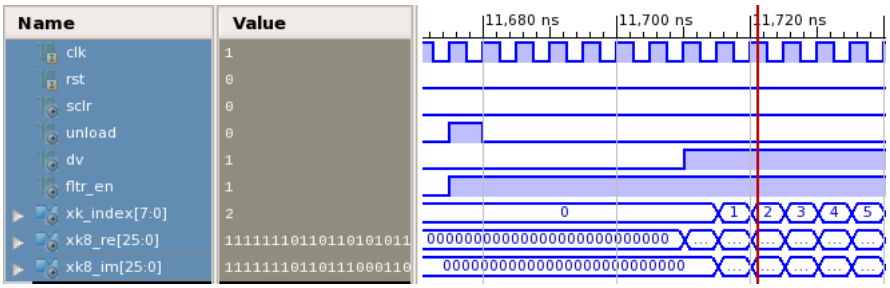


Figure 5.20: Filtering Initialisation.

After being activated, the filtering component starts receiving the Fourier domain angular projection in the *FLTR_RE* and *FLTR_IM* data ports, along with the *FLTR_ADDR*. For this example, the first interval of non-zero values starts at $FLTR\_INDEX = 80$.

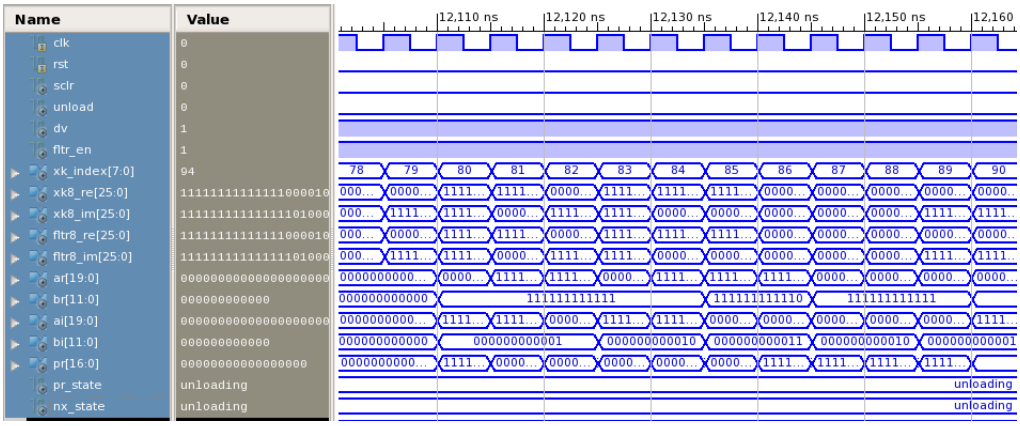


Figure 5.21: Filtering Execution.

Figure 5.21 shows how the complex multiplier operands *ar, ai, br* and *bi* perform the filtering, and how the result is given at the same clock cycle through *pr*. In the system, only the real part of the product is saved.
As mentioned before, the filtering is performed over the length+1 elements of the Fourier domain angular projection, so in this 256-length example, the filtering will stop its calculations at $FLTR\_INDEX = 129$.
Figure 5.22 show the real valued correspondence between the expected product MATLAB output (dotted line) and the output obtained from the VHDL system ISE ISim simulation (star symbols). This example belongs to the product between projection data in Fourier domain $R(s, 8)$ with the horizontal filter $W_{\psi}^{H}(s, 8)$, to produce the filtered projection (horizontal details at $\theta = 8$) $Q_{\psi}^{H}(s, 8)$.

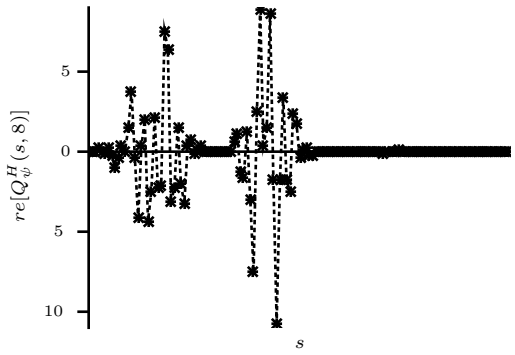


Figure 5.22: Positive Frequency Components of the Real Valued Horizontal Details $Re[Q_\psi^H(s,8)]$.

After filtering the complete 129-length sequence, the control component turns the FFT module to the setting-up and initialisation phase. After the filtering process, the FFT is configured with *fwd_inv_en* and *fwd_inv* to perform the inverse transform. The configuration sequence is shown in Figure 5.23.

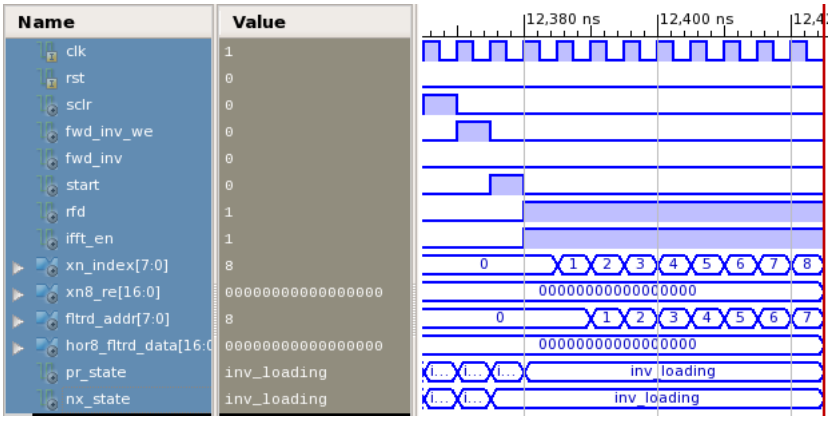


Figure 5.23: Inverse FFT Initialisation of Filtered Projection.

In order to perform the inverse FFT, the FFT module is fed with the obtained filtered data. At this stage, data belonging to the negative frequency spectrum is accomplished by reversing the order of the calculated positive frequency spectrum data stored values. Figure 5.24 show the full spectrum of horizontal details $Re[Q_\psi^H(s,8)]$.

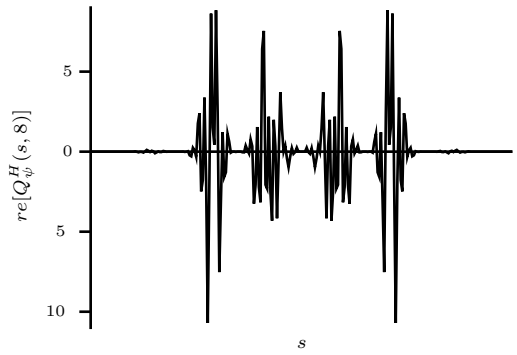


Figure 5.24: Real Valued Horizontal Details $Re[Q_\psi^H(s,8)]$.

The operation is exemplified in Figure 5.25, that show the symmetry at *fltrd_addr*= 129 during the data loading of the FFT module to perform the inverse FFT.

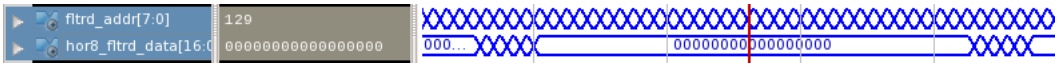


Figure 5.25: Reverse Order Filtered Data Loading.

Once the FFT module has been completely loaded, it will turn to the data processing phase. The consequent unloading phase must be performed by the backprojection component, which will require all Radon domain filtered data, to map the values over the Cartesian image coefficient grid. The resulting filtered projection data in Radon domain $Q_\psi^H(\rho,8)$ was previously shown in Figure 5.13, to illustrate the output obtained from the inverse FFT module.

5.2.2.4 FPGA Device Utilisation of the Wavelet-based Filtering of Projection Data

Table 5.6 show the resources utilisation output obtained from the ISE Wbepack synthesis report. Such a report was formulated when synthesising a system for the processing of 12 projection angles in parallel, as it is the maximum allowed by a single FFT module.

Logic Utilisation	Used	Available	Utilisation
Number of Slice Registers	24213	301440	8%
Number of Slice LUTs	24642	150720	16%
Number of fullt used LUT-FF pairs	16679	56299	51%
Number of bonded IOBs	2	600	0%
Number of Block RAM/FIFO	19	416	4%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of DSP48E1s	24	768	3%

Table 5.6: Filtering of Projection Data Resources Utilisation.

For the implementation of the filtering of projection data, it could be noticed that the most important hardware resources were the number of slice registers, slice LUTs, and block RAM/FIFO. Although some considerations were taken to reduce the number of block RAM, by storing only the necessary data to be processed, more logic that consumed slice registers had to be instantiated. Slice LUTs, as well as DSPs were mainly consumed by the FFT module and complex multipliers.
For the processing of projection data obtained at 45 equispaced angles, resources would increase in around four times, without considering a multiresolution decomposition. Such a consideration would consume around 32% of the available slice registers, 64% of available slice LUTs, and around 16% of block RAM. Such percentages would then be applied for a single resolution wavelet decomposition, and would increase by a factor of four for deeper resolutions. Therefore, the device resources would be totally consumed if a further resolution decomposition would be desired. Such statement is deduced according to the data truncation distribution of the multiresolution coefficient images, covered in Section 4.5.1 and illustrated in Figure 4.45.

5.2.3 Backprojection

The backprojection is the component that receives the filtered projection data, and its function is to map evrery value to its corresponding position within the Cartesian grid. To implement this component, it was opted for the same linear interpolation by which the MATLAB *iradon* function performs backprojection. The piece of code that performs backprojection is shown below.

```
1  T = X.*costheta(i) + Y.*sintheta(i);
2  A = floor(T);
3  IMG = IMG + (T-A).*t_proj(A+1+ctrIdx) + ...
       (A+1-T).*t_proj(A+ctrIdx);
```

where i is the index of the for loop that performs the operation for every angular projection, **X** and **Y** are arrays containing the Cartesian by which **T** indexes are calculated. **T** indexes give information about the position of every element within the projection data sequence. Array **A** is a rounded towards negative infinity, version of **T** and **IMG** is the pixel grid, accumulated at every angle to form the output image. The angular projection sequence is given as **t_proj**, and *ctrIdx* is the position weight to obtain the true centre value within the **t_proj** sequence.
As it can be seen, all the values, except for **IMG**, are known, so Arrays **T**, **A**, and the weight *ctrIdx* can be calculated and introduced as numerical values. Therefore it is possible to split the calculation of **IMG** into three parts:

$$\begin{aligned} \mathbf{IMG_1} &= (\mathbf{T} - \mathbf{A}). * \mathbf{t\_proj}(\mathbf{A} + ctrIdx + 1) \\ \mathbf{IMG_2} &= (\mathbf{A} - \mathbf{T} + 1). * \mathbf{t\_proj}(\mathbf{A} + ctrIdx) \\ \mathbf{IMG} &= \mathbf{IMG_1} + \mathbf{IMG_2} \end{aligned} \tag{5.1}$$

In MATLAB it is possible to perform array indexing but in this VHDL system, it is needed to evaluate value by value as they come out from the inverse FFT calculation. The index with its corresponding data value are the inputs to the backprojection component shown in Figure 5.26.

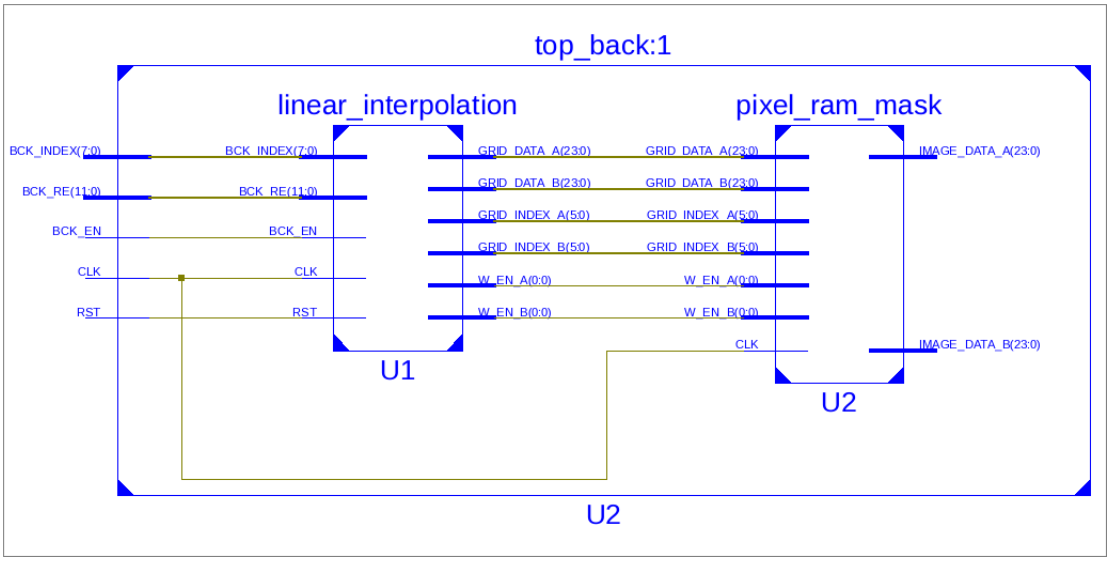


Figure 5.26: Backprojection Component.

The port description for the backprojection is given in Table 5.7

Port Name	Direction	Description
BCK_INDEX	Input	Filtered Projection Index
BCK_RE	Input	Filtered Projection Value
BCK_EN	Input	Backprojection Enable
IMAGE_DATA_A	Output	Calculated Image A
IMAGE_DATA_B	Output	Calculated Image B

Table 5.7: Backprojection Ports Description.

This way, according to a comparison between the input *BCK_INDEX* values and precomputed indexes $(\mathbf{A} + ctrIdx + 1)$ and $(\mathbf{A} + ctrIdx)$, the input data value *BCK_RE* can be multiplied by its corresponding linear interpolation weights $(\mathbf{T} - \mathbf{A})$ for $\mathbf{IMG_1}$, and $(\mathbf{A} - \mathbf{T} + 1)$ for $\mathbf{IMG_1}$.
This is easily implemented through a LUT containing both $(\mathbf{A} + ctrIdx)$ and $(\mathbf{A} + ctrIdx + 1)$, with its corresponding weight values to build $\mathbf{IMG_1}$ and $\mathbf{IMG_2}$.
In VHDL, the LUT is implemented through two processes, each one containing *case* statements to compare the input (BCK_INDEX) along with $(\mathbf{A} + ctrIdx)$ in the first process, and with $(\mathbf{A} + ctrIdx + 1)$ in the second process. A particular observation is that $(\mathbf{A} + ctrIdx)$ is a delayed value of $(\mathbf{A} + ctrIdx + 1)$, so when *BCK_INDEX* matches with $(\mathbf{A} + ctrIdx)$, its corresponding product with $(\mathbf{A} - \mathbf{T} + 1)$ is calculated and the result stored. At

the next iteration *BCK_INDEX* will match with $(\mathbf{A} + ctrIdx + 1)$, the corresponding product with $(\mathbf{T} - \mathbf{A})$ calculated and the result accumulated with the stored value. The process is repeated until matching all filtered projection data values.
In the complete system, backprojection output values are sorted into row RAMs where every value is accumulated at the input, in order to obtain the final Cartesian pixel values that comprise the coefficient image.

5.2.3.1 FPGA Device Utilisation of the Backprojection

Table 5.8 show the resources utilisation output obtained, again from the ISE Wbepack synthesis report, for the backprojection component.

Logic Utilisation	Used	Available	Utilisation
Number of Slice Registers	5820	301440	1%
Number of Slice LUTs	20900	150720	13%
Number of fullt used LUT-FF pairs	2125	56299	8%
Number of bonded IOBs	221	600	36%
Number of Block RAM/FIFO	1	416	0%
Number of BUFG/BUFGCTRLs	2	32	6%

Table 5.8: Backprojection Resources Utilisation.

For the implementation of the backprojection, as it is mainly implemented as a LUT, the main consumed logic belong to the slice LUTs. The resources utilisation of this component is not as critical as the filtering of projection data, as it will require around 13% of slice LUTs for every resolution, considering a 64 by 64 pixel input phantom.

5.2.4 Parallel Inverse Two-Dimensional Wavelet Transform

This component works with the row RAMs generated through the backprojection component. The inverse FWT is virtually implemented in the same way as it is defined in the parallel synthesis QMF of Figure 4.38. To do so, two instantiated IP core Finite Impulse Response (FIR) filters were employed, each one corresponding to the high pass and low pass reverse filters.
During its configuration of the FIR modules, filter coefficients can either be loaded through the core instantiation Graphical User Interface (GUI), or

through a memory generator *.coe* file. The FIRs were chosen to be instatiated as single rate filters with the input sample period of the ML605 Virtex board 200MHz clock, and with a simple Multiply-Accumulate (MAC) architecture optimised for area resources. The FIR modules allow up to 16 parallel input paths. More information about the configuration and functionality of the FIR LogiCore IP can be found in [151].

For the created example, a set of 8 by 8-pixel input coefficient images were employed, which according to the backprojection input format, is arranged into eight row ROMs, each one containing eight elements. So by exploiting the 16 parallel input paths that the FIR modules allow, filtering can be fully implemented in parallel for the rows filtering, by interpolating zero values according to the upsampling factor. Conversely, for the columns filtering, input data has at least doubled its size. This behaviour limits the parallel implementation of the inverse FWT, as in order to save resources, filtering must be partially sequenced.

The functionality of the inverse two-dimensional FWT, consists therefore in the proper manipulation of the input/output data values of the FIR modules.

The documentation of the inverse FWT is limited to the high amount of data that is employed, either in the schematics, as in the simulator results. As a last resource to show evidence of the implementation, the VHDL behavioural hierarchy is shown in the next figure.

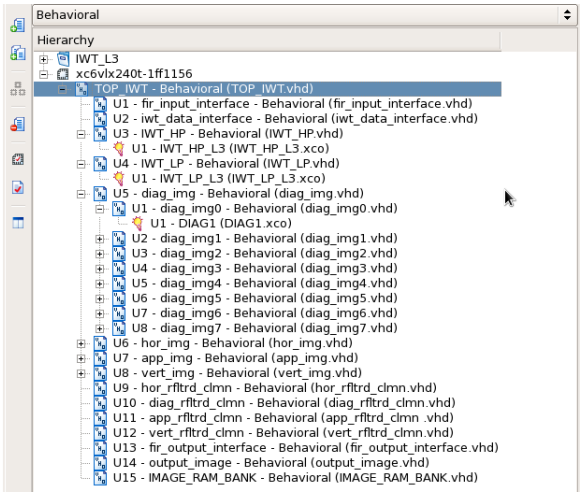


Figure 5.27: Inverse Two-Dimensional FWT Behavioural Hierarchy.

5.2.4.1 FPGA Device Utilisation of the Parallel Inverse Two-Dimensional Wavelet Transform

Table 5.9 show the resources utilisation output obtained, once again from the ISE Wbepack synthesis report, for the Parallel Inverse Two-Dimensional Wavelet Transform.

Logic Utilisation	Used	Available	Utilisation
Number of Slice Registers	40064	301440	13%
Number of Slice LUTs	19679	150720	13%
Number of fullt used LUT-FF pairs	3444	56299	6%
Number of bonded IOBs	308	600	51%
Number of Block RAM/FIFO	16	416	3%
Number of BUFG/BUFGCTRLs	1	32	3%
Number of DSP48E1s	96	768	12%

Table 5.9: Parallel Inverse Two-Dimensional Wavelet Transform Resources Utilisation.

The main component of the inverse two-dimensional WT was the FIR module, whose operation require the utilisation of DSP resources. In contrast to the filtering of projection data, as well as the backprojection, only one inverse two-dimensional WT is required in a complete system. The concern with the integration of the inverse two-dimensional WT into a single system, along with the filtering of projection data and backprojection, is the utilisation of DSP resources that should have to be shared with the FFT modules.

5.3 Summary

An FPGA-based system approach was developed with the aim to test with the parallel functionality of the high-level MATLAB designed algorithm. The implementation comprises the processing of a single angular projection, which represents the smaller processing unit that can be replicated to integrate a full system. Moreover, the system was designed through a modular form to be easily adapted to the application requirements.
The FPGA system was split into three main components, the filtering of projection data, the backprojection, and the the inverse two-dimensional FWT. The filtering of projection data is constituted by a ROM containing projection data, a FFT LogicCORE IP that performs both forward and inverse transforms,

a complex multiplier that performs the Fourier domain filtering, a control unit and input output interfaces.
The backprojection was formulated through a LUT from which filtered data indexes are analysed in order to locate theme into its correspondent Cartesian location.
The inverse two-dimensional FWT is essentially implemented as a replica of the parallel QMF. It performs filtering in the space domain through a pair of instantiated FIR filters, which perform the filtering along both the rows and columns of the input image coefficients data. Because of resources limitations, the inverse two-dimensional FWT can not be totally implemented in parallel, as the FIR modules are limited to 16 parallel input paths. More than two FIR modules can be instantiated, but at the high expense of resources.
In this thesis, it was only possible to show results about the formulation and functionality of the hardware implementation design. Although the off-the-shelf FPGA evaluation board is one of the high-end Xilinx devices, it resulted inefficient in terms of on-chip storage and DSP resources for the integration of a system comprising the parallel block reconstruction of the complete wavelet-based, parallel block multiresolution algorithm.

Chapter 6

Conclusions and Future Work

The background theory covered in the introductory chapters of this thesis, allowed the acquisition of the appropriate knowledge that was used to approach the main challenge during the development of this research: to converge both Tomography and the WT, into a fast parallel image reconstruction algorithm. Since the early years of its creation, the WT has been applied to Tomography with the aim of reducing the radiation dose exposure to patients, through the accurate image reconstruction from truncated projection data. Although no evidence of an application, properly dedicated to the achievement of fast image reconstructions within a parallel framework, was found in the literature.
The algorithm proposed in this thesis employed the WT as a tool to split the Tomography inverse problem into smaller and less complex tasks, that when executed in parallel, could achieve higher speed performance without compromising the output image quality. In this regard, the WT capabilities were exploited to deliver a parallel multiresolution representation. The main contribution in this thesis, is the design of a wavelet-based, parallel block multiresolution image reconstruction algorithm, from which a hardware implementation approach was formulated.
Preliminary results of the developed algorithm, allowed for a presentation in the PHOTON14 Conference, and further results have been prepared and deposited for submission in IEEE Sensors Journal Manuscript Central portal. The draft manuscript has been included in Appendix B.
Through this chapter, the most relevant features of the fast parallel reconstruction design, as well as it hardware implementation approach, are summarised. The chapter finalises by exposing some suggested future work.

6.1 Conclusions

The main motivation for the development of this research, was the achievement of a parallel fast reconstruction algorithm, inspired by the wavelet-based available research, related to the accurate reconstruction of reduced-area images, from truncated projection data. This was achieved through the development of the wavelet-based, parallel block multiresolution algorithm, which not only exploited the already demonstrated WT capabilities to achieve accurate images from truncated data, but also generalised them to a parallel multiresolution framework, that allowed faster image reconstructions.
The proposed algorithm is based on the FBP, whose computer implementation broadly established the beginning of this research development. Being able to achieve the computer implementation of the FBP, allowed the better understanding of its involved fundamental concepts, through a more practical framework. The computer implementation of the FBP also made it possible to gain experience with its structure, which triggered the interest in customising the algorithm according to the project objectives.
By the customisation of the FBP, it was possible to understand the effects caused by the two-dimensional FWT incorporation, which according to the reference literature, its repercussion is addressed at the filtering stage of the FBP. The WT, via its time-frequency localisation properties, allows projection data to remain compactly supported after being filtered with the ramp weighting function. The implication of such feature relies in the fact that projection data becomes globally dependent upon the total line integral measurements, as its support is spread after being ramp-filtered [27].
Through the addition of the two-dimensional FWT into the filtering stage of the FBP, it was possible to realize that, according to the way in which the wavelet and scaling functions tile the Fourier space, the computation load of the FWT output coefficients, could be reduced by discarding redundant data. This was partially true, as the computation of the approximations coefficient involves data from all the angular projections, within the Fourier space. Such a situation, attracted the interest to analyse Tomography projection data, through the multiresolution scheme of the two-dimensional FWT. Unfortunately, such approach had to be discarded almost by default, as the computation of the FWT output coefficient images is performed through a recursive procedure. Further research on the topic, drove the attention to a different alternative that

made it possible to overcome the inherent recursive computation of the multiresolution FWT. Through the multirate Nobel identities, a parallel computation framework, in the Fourier domain, could be achieved for the computation of the multiresolution two-dimensional FWT. Decomposing projection data into a set of different frequency scale components, considerably reduced the reconstruction time, by assuming a concurrent computation of every coefficient image.
For the achievement of the block-decomposition generalisation, the multiresolution parallel scheme was also determinant in terms of quality, as it allowed to bypass the need to truncate the approximations coefficient image, which involves data from all projections at all angles. The initial objective of the WT utilisation was the accurate reconstruction from truncated data, although, for the benefit of this work, some other features arose, which made it possible to extend the development of the algorithm to a higher grade of parallelism.
For the truncation of projection data, a simple methodology was developed, with the aim to ease the process of extracting the line integral values, corresponding to block supports within the FOV of the projected object. The distribution, as well as the size of the block tiles, was determined according to the multiresolution properties of the FWT analysis of projection data.
The algorithm was tested at every stage of its development, with synthetic data obtained from the RT of simulated phantoms, mostly by the 256 by 256 Shepp Logan phantom. Time measurements were performed with the *tic/tac* MATLAB function, for which in order to enhance its precision, a single processor core was employed. Time measurements were focused in measuring the backprojection for the standard FBP and the wavelet-based algorithms. For wavelet-based algorithms, the time spent in the WT synthesis was also considered. To have an estimate of the quality accuracy of the reconstructed images, the pixel grey-scale intensity values was compared between the input phantom and the reconstructed image.
Small differences were encountered between the reconstructed image and the input phantom, so the reconstruction images showed to be an accurate estimate of the input phantom.
In terms of speed, the parallel block decomposition approach, achieved backprojection and FWT at around 48 times faster than the backprojection of the standard FBP.

In order to test with a lower-level implementation of the MATLAB designed algorithm, an FPGA-based system approach was developed, with the objective to overcome the MATLAB inherent sequential limitation and analyse the functionality of the algorithm, through a true parallel computation platform. As part of the design process, the implementation was limited to the processing of a single angular projection, which represents the smaller processing unit within the algorithm. The system was designed in a modular form, so the single unit processing can be adapted to the user requirements, as well as replicated up to the application needs.
The FPGA system was split into three main components, the filtering of projection data, which comprised the filtering in Fourier domain between wavelet-modified filters and projection angles. The output of the filtering component is the filtered data in Radon domain, read to be backprojected. The second component is the Backprojection, and the third component is the inverse two-dimensional FWT.
In this thesis, it was only possible to show results about the formulation and functionality of the hardware implementation design. Although the off-the-shelf FPGA evaluation board is one of the high-end Xilinx devices, it resulted inefficient in terms of on-chip storage and DSP resources for the integration of a system comprising the parallel block reconstruction of the 3-scale decomposition 256 by 256 Shepp Logan phantom. A different integration of the system to accomplish integration within the same board is achievable, although parallelism must be limited up to a certain degree, which would not match with the parallelism degree that we were looking for, in the development of the high-level MATLAB algorithm. Unfortunately, looking for for a different alternative, that would be able to accomplish the true parallel system integration, was already outside the allowed time to accomplish with the stipulated deadline.

6.2 Future Work

The algorithm presented in this thesis has been designed by considering only projection data acquired from a parallel geometry, therefore future work can be performed by expanding the algorithm capabilities to the processing of projection data acquired from a fan-beam geometry. Such framework employed in wavalet-based FBP algorithms is not new and has been demonstrated to be

feasible [152, 153].
A more challenging approach should be to extend the algorithm capabilities to three dimensions, either in the parallel or cone-beam geometries. Some developments of three-dimensional Tomography reconstruction by employing the WT have been reported in the last decade [80, 154–156].
In terms of the hardware implementation, the most evident work to follow-up in the development of the fast parallel reconstruction method presented in this thesis is the true parallel implementation of a full integrated system. As previously mentioned in the conclusions section, the off-the-shelf evaluation board lacked of the enough on-chip storage and DSP resources, to accomplish the full system integration. More advanced FPGA devices, belonging to the Xilink ultrascale virtex range, are already available with the so-called *3D IC* system integration with up to 4.4 million logic cells. This kind of devices are every time looking forward to achieve the same capabilities of ASICs, and consequently into higher costs [157].
A different approach could consist in the design of a custom FPGA-based processing board, comprising several lower-range (i.e. affordable) devices capable to perform the required amount of parallel computations. A similar device is the Siemens ImageProX board, which has nine Virtex-4 FPGAs, with eight of them dedicated to processing and one as the control processing and interface unit [55].
Through this assumptions, it may be inferred that the Wavelet-Based, Multiresolution Parallel Block Reconstruction Algorithm can be implemented with the already available existing technologies. Either by a custom FPGA-based board at reasonable costs, or with the state-of-the-art single-chip alternatives, which are too expensive.

References

[1] A. Kak and M. Slaney, *Principles of computerized tomographic imaging*. Classics in applied mathematics, Society for Industrial and Applied Mathematics, 2001.

[2] F. Natterer, *The Mathematics of Computerized Tomography*. Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 2001.

[3] J. Radon, "On the determination of functions from their integral values along certain manifolds," *Medical Imaging, IEEE Transactions on*, vol. 5, pp. 170–176, Dec 1986.

[4] A. M. Cormack, "Representation of a Function by Its Line Integrals, with Some Radiological Applications," *Journal of Applied Physics*, vol. 34, p. 2722, June 1963.

[5] G. Hounsfield, "Apparatus for examining a body by radiation such as X or gamma radiation," *US Patent 3,944,833*, 1976.

[6] R. A. Robb, "X-ray computed tomography: an engineering synthesis of multiscientific principles.," *Critical reviews in biomedical engineering*, vol. 7, pp. 265–333, Jan. 1982.

[7] T. York, "Status of electrical tomography in industrial applications," *Journal of Electronic Imaging*, vol. 10, no. 3, pp. 608–619, 2001.

[8] P. C. Seynaeve and J. I. Broos, "The history of tomography.," *Journal belge de radiologie*, vol. 78, pp. 284–8, Oct. 1995.

[9] G. T. Herman, "Image reconstruction from projections," *Real-Time Imaging*, vol. 1, pp. 3–18, Apr. 1995.

[10] J. Hsieh, *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*. Spie Press Monograph . Pm114, Society of Photo Optical, 2003.

[11] F. Natterer and F. Wûbbeling, *Mathematical Methods in Image Reconstruction*. Siam Monographs on Mathematical Modeling and Computation, Society for Industrial and Applied Mathematics, 2001.

[12] K. J. Batenburg and L. Plantagie, "Fast approximation of algebraic reconstruction methods for tomography.," *IEEE transactions on image processing*, vol. 21, pp. 3648–58, Aug. 2012.

[13] F. E. Boas and D. Fleischmann, "CT artifacts: causes and reduction techniques," *Imaging in Medicine*, vol. 4, pp. 229–240, Apr. 2012.

[14] A. K. Louis and A. Rieder, "Incomplete data problems in X-ray computerized tomography," vol. 56, no. 4, pp. 371–383, 1989.

[15] R. Perry, "On reconstructing a function on the exterior of a disk from its Radon transform," *Journal of Mathematical Analysis and Applications*, vol. 59, no. 2, pp. 324–341, 1977.

[16] R. Clackdoyle and M. Defrise, "Region-of-interest reconstruction from incomplete data," *IEEE Signal Processing Magazine*, no. July, pp. 60–80, 2010.

[17] A. Bilgot, L. Desbat, and V. Perrier, "Filtered backprojection method and the interior problem in 2D tomography." `https://hal.archives-ouvertes.fr/hal-00591849/`, Sept. 2009.

[18] G. Wang, H. Yu, and B. De Man, "An outlook on x-ray CT research and development," *Medical Physics*, vol. 35, no. 3, p. 1051, 2008.

[19] G. Wang and H. Yu, "The meaning of interior tomography," *Physics in Medicine and Biology*, vol. 58, no. 16, p. R161, 2013.

[20] E. T. Quinto, "Singularities of the X-Ray Transform and Limited Data Tomography in R$\hat{2}$ and R$\hat{3}$," *SIAM Journal on Mathematical Analysis*, vol. 24, no. 5, pp. 1215–1225, 1993.

[21] A. Faridani, "Local tomography," *SIAM Journal on Applied Mathematics*, vol. 52, no. 2, pp. 459–484, 1992.

[22] P. Maass, "The Interior Radon Transform," *SIAM Journal on Applied Mathematics*, vol. 52, no. 3, pp. 710–724, 1992.

[23] J. DeStefano and T. Olson, "Wavelet localization of the radon transform in even dimensions," in *Time-Frequency and Time-Scale Analysis, 1992., Proceedings of the IEEE-SP International Symposium*, pp. 137–140, Oct 1992.

[24] G. Wang, "The meaning of interior tomography," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 5764–5767, 2011.

[25] T. M. Lehmann, T. Aach, and H. Witte, "Sensor, signal and image informatics - state of the art and current topics," *IMIA Yearbook 2006: Assessing Information - Technologies for Health*, vol. 1, no. 1, pp. 57–67, 2006.

[26] R. Azencott, A. Sen, B. G. Bodmann, K. C. Li, D. Labate, and X. Zhou, "Searchlight CT: A new reconstruction method for collimated X-ray tomography," *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, 2012.

[27] T. Olson and J. DeStefano, "Wavelet localization of the Radon transform," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 2055–2067, 1994.

[28] A. I. Katsevich and A. G. Ramm, "Pseudolocal tomography," *SIAM Journal on Applied Mathematics*, vol. 56, no. 1, pp. pp. 167–191, 1996.

[29] F. Noo, R. Clackdoyle, and J. D. Pack, "A two-step Hilbert transform method for 2D image reconstruction.," *Physics in medicine and biology*, vol. 49, no. 17, pp. 3903–3923, 2004.

[30] M. Holschneider, "Inverse Radon transforms through inverse wavelet transforms," *Inverse problems*, vol. 853, 1991.

[31] C. A. Berenstein and D. F. Walnut, "Wavelets and local tomography," in *Wavelets in Medicine and Biology*, pp. 231–262, CRC Press, 1995.

[32] A. E. Yagle, "Region-of-interest tomography using the wavelet transform and angular harmonics," *IEEE Signal Processing Letters*, vol. 1, pp. 134–135, Sept. 1994.

[33] A. H. Delaney and Y. Bresler, "Multiresolution tomographic reconstruction using wavelets," *IEEE Transactions on Image Processing*, vol. 4, pp. 799–813, Jan. 1995.

[34] F. Rashid-Farrokhi, K. R. Liu, C. a. Berenstein, and D. Walnut, "Wavelet-based multiresolution local tomography.," *IEEE transactions on image processing*, vol. 6, pp. 1412–30, Jan. 1997.

[35] F. Truchetet and O. Laligant, "Review of industrial applications of wavelet and multiresolution-based signal and image processing," *Journal of Electronic Imaging*, vol. 17, p. 031102, July 2008.

[36] I. M. Dremin, O. V. Ivanov, and V. a. Nechitailo, "Wavelets and their use," *Physics-Uspekhi*, vol. 44, p. 63, May 2001.

[37] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, pp. 674–693, Jul 1989.

[38] B. Hubbard, *The world according to wavelets: the story of a mathematical technique in the making*. Ak Peters Series, A.K. Peters, 1998.

[39] F. Peyrin, M. Zaim, and R. Goutte, "Multiscale reconstruction of tomographic images," *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, no. 33, pp. 0–3, 1992.

[40] S. Basu and Y. Bresler, "O(N(2)log(2)N) filtered backprojection reconstruction algorithm for tomography.," *IEEE transactions on image processing*, vol. 9, no. 10, pp. 1760–1773, 2000.

[41] T. Rodet, L. Desbat, and P. Grangeat, "Parallel algorithm based on a frequential decomposition for dynamic 3d computed tomography," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 7 pp.–, 2003.

[42] Y. O'Connor and J. Fessler, "Fourier-based forward and back-projectors in iterative fan-beam tomographic image reconstruction," *Medical Imaging, IEEE Transactions on*, vol. 25, pp. 582–589, May 2006.

[43] S. Xiao, Y. Bresler, and J. Munson, D.C., "O(n2 log n) native fan-beam tomographic reconstruction," in *Biomedical Imaging, 2002. Proceedings. 2002 IEEE International Symposium on*, pp. 824–827, 2002.

[44] Y. Bresler and J. Brokish, "A hierarchical algorithm for fast backprojection in helical cone-beam tomography," *2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821)*, no. MAY 2004, 2004.

[45] A. K. George and Y. Bresler, "Shear-based fast hierarchical backprojection for parallel-beam tomography," *IEEE Transactions on Medical Imaging*, vol. 26, no. 3, pp. 317–334, 2007.

[46] A. George and Y. Bresler, "Fast tomographic reconstruction via rotation-based hierarchical backprojection," *SIAM Journal on Applied Mathematics*, vol. 68, no. 2, pp. 574–597, 2007.

[47] M. Ingerhed, *Fast Backprojection in Computed Tomography: Implementation and Evaluation*. LiU-Tek-Lic, Department of Electrical Engineering, Linköpings Universitet, 1999.

[48] S. Nilsson and L. E. Andersson, "Application of fast back-projection techniques for some inverse problems of synthetic aperture radar," *Proc. SPIE*, vol. 3370, pp. 62–72, 1998.

[49] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions.," *Physics in medicine and biology*, vol. 49, no. 11, pp. 2463–2475, 2004.

[50] C. Bobda, *Introduction to reconfigurable computing: Architectures, algorithms, and applications*. 2008.

[51] "TeraRecon, Inc." `http://www.terarecon.com/`.

[52] "Varian Medical Systems, Inc." `http://www.varian.com/`.

[53] "Mercury Systems, Inc." `http://www.mrcy.com/`.

[54] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Transactions on Nuclear Science*, vol. 52, no. 3 I, pp. 654–663, 2005.

[55] H. Scherl, M. Kowarschik, H. G. Hofmann, B. Keck, and J. Hornegger, "Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction," *Parallel Computing*, vol. 38, no. 3, pp. 111–124, 2012.

[56] A. Myagotin, A. Voropaev, L. Helfen, D. Hanschke, and T. Baumbach, "Efficient volume reconstruction for parallel-beam computed laminography by filtered backprojection on multi-core clusters," *Image Processing, IEEE Transactions on*, vol. 22, pp. 5348–5361, Dec 2013.

[57] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, "Medical image processing on the GPU - Past, present and future," 2013.

[58] U. J. Kapasi, S. Rixner, W. J. Dally, B. Khailany, J. H. Ahn, P. Mattson, and J. D. Owens, "Programmable stream processors," *Computer*, vol. 36, no. 8, pp. 54–62, 2003.

[59] K. Mueller, "Use of graphics hardware to accelerate algebraic reconstruction methods," *Proceedings of SPIE*, no. 614, pp. 615–625, 1999.

[60] D. Stsepankou, K. Kommesser, J. Hesser, and R. Manner, "Real-time 3D cone beam reconstruction," *IEEE Symposium Conference Record Nuclear Science 2004.*, vol. 6, no. C, pp. 3648–3652, 2004.

[61] M. Churchill, G. Pope, J. Penman, D. Riabkov, X. Xue, and a. Cheryauka, "Hardware-accelerated cone-beam reconstruction on a mobile C-arm," *Proceddings SPIE*, vol. 6510, pp. 65105S–65105S–8, Mar. 2007.

[62] M. Knaup, S. Steckmann, O. Bockenbach, and M. Kachelrieß, "Tomographic image reconstruction using the cell broadband engine (CBE) general purpose hardware," *Proceedings of SPIE*, vol. 6498, pp. 64980P–64980P–10, 2007.

[63] N. Neophytou, F. Xu, and K. Mueller, "Hardware acceleration vs. algorithmic acceleration: Can GPU-based processing beat complexity optimization for CT," *SPIE Medical Imaging*, vol. 6510, pp. 65105F–65105F–9, 2007.

[64] T. Callahan and J. Wawrzynek, "Instruction-level parallelism for reconfigurable computing," in *Field-Programmable Logic and Applications From FPGAs to Computing Paradigm* (R. Hartenstein and A. Keevallik, eds.), vol. 1482 of *Lecture Notes in Computer Science*, pp. 248–257, Springer Berlin Heidelberg, 1998.

[65] "FPGA vs. ASIC." `http://www.xilinx.com/fpga/asic.htm`.

[66] M. Trepanier and I. Goddard, "Adjunct processors in embedded medical imaging systems," *Proc. SPIE*, vol. 4681, pp. 416–424, 2002.

[67] M. Leeser, S. Coric, E. Miller, H. Yu, and M. Trepanier, "Parallel-beam backprojection: An FPGA implementation optimized for medical imaging," *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, vol. 39, no. 3, pp. 295–311, 2005.

[68] I. Goddard and M. Trepanier, "High-speed cone-beam reconstruction : an embedded systems approach," *Communications*, vol. 1, no. 978, pp. 1–9, 2002.

[69] D. Stsepankou, K. Kornmesser, J. Hesser, and R. Manner, "Fpga-acceleration of cone-beam reconstruction for the x-ray ct," in *Field-Programmable Technology, 2004. Proceedings. 2004 IEEE International Conference on*, pp. 327–330, Dec 2004.

[70] J. Brokish and Y. Bresler, "Combined algorithmic and hardware acceleration for ultra-fast backprojection," *IEEE Nuclear Science Symposium Conference Record*, vol. 5, pp. 3915–3918, 2007.

[71] P. Basu and M. Manjunatha, "VHDL modelling and simulation of parallel-beam filtered backprojection for ct image reconstruction," *2009 International Multimedia, Signal Processing and Communication Technologies, IMPACT 2009*, no. 1, pp. 213–216, 2009.

[72] L. Maltar, F. Franca, V. Alves, and C. Amorim, "Reconfigurable hardware for tomographic processing," *Proceedings. XI Brazilian Symposium on Integrated Circuit Design (Cat. No.98EX216)*, pp. 4–9, 1998.

[73] J. K. Kim, J. a. Fessler, and Z. Zhang, "Forward-Projection Architecture for Fast Iterative Image Reconstruction in X-ray CT.," *IEEE transactions*

on signal processing : a publication of the IEEE Signal Processing Society, vol. 60, pp. 5508–5518, Oct. 2012.

[74] H. H. Szu, J. T. DeWitte, Jr., J. P. Garcia, B. A. Telfer, T. E. Olson, D. M. Healy, Jr., and R. J. P. de Figueiredo, "Wavelet transform for local tomography reconstruction," *Proc. SPIE*, vol. 2491, pp. 794–818, 1995.

[75] M. Haghnegahdar, S. Boden, and U. Hampel, "Investigation of mass transfer in milli-channels using high-resolution microfocus x-ray imaging," *International Journal of Heat and Mass Transfer*, vol. 93, pp. 653 – 664, 2016.

[76] M. Schubert, A. Bieberle, F. Barthel, S. Boden, and U. Hampel, "Advanced tomographic techniques for flow imaging in columns with flow distribution packings," *Chemie Ingenieur Technik*, vol. 83, no. 7, pp. 979–991, 2011.

[77] H. V. Hristov, B. Stephan, H. Uwe, K. Holger, H. Gnther, and S. Wilfried, "A study on the two-phase flow in a stirred tank reactor agitated by a gas-inducing turbine," *Chemical Engineering Research and Design*, vol. 86, no. 1, pp. 75 – 81, 2008.

[78] M. Costin, D. Lazaro-Ponthus, S. Legoupil, P. Duvauchelle, and V. Kaftandjian, "A 2D multiresolution image reconstruction method in X-ray computed tomography.," *Journal of X-ray science and technology*, vol. 19, pp. 229–47, Jan. 2011.

[79] L. Li, H. Toda, T. Ohgaki, M. Kobayashi, T. Kobayashi, K. Uesugi, and Y. Suzuki, "Wavelet-based local region-of-interest reconstruction for synchrotron radiation x-ray microtomography," *Journal of Applied Physics*, vol. 102, no. 11, 2007.

[80] X. Yin, B.-H. Ng, B. Ferguson, S. Mickan, and D. Abbott, "2-d wavelet segmentation in 3-d t-ray tomography," *Sensors Journal, IEEE*, vol. 7, pp. 342–343, March 2007.

[81] X. Yin, B. W.-H. Ng, B. Ferguson, and D. Abbott, "Wavelet based local tomographic image using terahertz techniques," *Digital Signal Processing*, vol. 19, pp. 750–763, July 2009.

[82] X. Yin, B.-H. Ng, J. Zeitler, K. L. Nguyen, L. Gladden, and D. Abbott, "Local computed tomography using a thz quantum cascade laser," *Sensors Journal, IEEE*, vol. 10, pp. 1718–1731, Nov 2010.

[83] R. N. Bracewell and A. C. Riddle, "Inversion of Fan-Beam Scans in Radio Astronomy.," *Astrophysical Journal*, vol. 150, p. 427, Nov. 1967.

[84] M. Defrise and G. T. Gullberg, "Image reconstruction.," *Physics in medicine and biology*, vol. 51, pp. R139–54, July 2006.

[85] L. Chen, *Digital and Discrete Geometry: Theory and Algorithms*. SpringerLink : Bücher, Springer International Publishing, 2014.

[86] T. Buzug, *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT*. Springer, 2008.

[87] P. Boccacci, *Introduction to Inverse Problems in Imaging*. CRC Press, 1998.

[88] G. Zeng, *Medical Image Reconstruction: A Conceptual Tutorial*. Springer, 2010.

[89] R. Bracewell, *The Fourier Transform and Its Applications*. Electrical engineering series, McGraw Hill, 2000.

[90] A. Devaney, "A filtered backpropagation algorithm for diffraction tomography," *Ultrasonic Imaging*, vol. 4, no. 4, pp. 336 – 350, 1982.

[91] R. Lewitt, "Reconstruction algorithms: Transform methods," *Proceedings of the IEEE*, vol. 71, no. 3, pp. 390–408, 1983.

[92] E. Kreyszig, *Advanced Engineering Mathematics*. John Wiley & Sons, 2010.

[93] H. Stark, J. Woods, I. Paul, and R. Hingorani, "Direct fourier reconstruction in computer tomography," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 29, pp. 237–245, Apr 1981.

[94] M. Defrise, F. Noo, R. Clackdoyle, and H. Kudo, "Truncated hilbert transform and image reconstruction from limited tomographic data," *Inverse Problems*, vol. 22, no. 3, p. 1037, 2006.

[95] Y. Ye, H. Yu, Y. Wei, and G. Wang, "A general local reconstruction approach based on a truncated hilbert transform," *Journal of Biomedical Imaging*, vol. 2007, pp. 2–2, Jan. 2007.

[96] S. R. Cherry, "Multimodality imaging: Beyond pet/ct and spect/ct," *Seminars in Nuclear Medicine*, vol. 39, no. 5, 2009.

[97] D. Gabor, "Theory of communication," *Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering*, vol. 93, no. 26, pp. 429–441, 1946.

[98] J. Allen, "Applications of the short time fourier transform to speech processing and spectral analysis," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '82.*, vol. 7, pp. 1012–1015, May 1982.

[99] "Local time-frequency analysis and short time Fourier transform." `http://www.math.ucdavis.edu/~strohmer/research/gabor/gaborintro/node3.html`.

[100] R. Merry and M. Steinbuch, "Wavelet theory and applications." literature study, Eindhoven university of technology, Department of mechanical engineering, Control systems technology group, 2005.

[101] A. Haar, "Zur Theorie der orthogonalen Funktionensysteme," *Mathematische Annalen*, vol. 71, no. 1, pp. 38–53, 1911.

[102] J. E. Littlewood and R. E. A. C. Paley, "Theorems on fourier series and power series," *Journal of the London Mathematical Society*, vol. s1-6, no. 3, pp. 230–233, 1931.

[103] K. G. Wilson, "The renormalization group: Critical phenomena and the Kondo problem," vol. 47, no. 4, pp. 773–840, 1975.

[104] D. Marr, *Vision: A computational approach*. Freeman & Co., San Francisco, 1982.

[105] A. Croisier, D. Esteban, and C. Galand, "Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques," in *International Conference on Information Sciences and Systems*, vol. 2, pp. 443–446, Patras, Greece, 1976.

[106] F. Mintzer, "Filters for distortion-free two-band multirate filter banks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 33, no. 3, pp. 626–630, 1985.

[107] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Magazine*, vol. 8, no. 4, pp. 14–38, 1991.

[108] D. E. Barrick and W. H. Peake, "Scattering From Surfaces With Different Roughness Scales; Analysis and Intrepretation," Tech. Rep. September 1967, Electroscience Lab., 1967.

[109] Dag Tollefsrud (Norwegian Geotechnical Institute), "Seismics and EM Complement each other." `http://www.ngi.no/en/Selected-topics/EM-technology-in-ocean-depths/Seismics-and-EM-complement-each-other/`.

[110] J. Morlet, "Wave propagation and sampling theoryPart I: Complex signal and scattering in multilayered media," *Geophysics*, vol. 47, no. 2, p. 203, 1982.

[111] J. Morlet, G. Arensz, E. Fourgeau, and D. Giard, "Wave propagation and sampling theory: Sampling theory and complex waves," *Geophysics*, vol. 41, no. 2, pp. 222–236, 1982.

[112] I. Daubechies, "Where do wavelets come from? - a personal point of view," *Proceedings of the IEEE*, vol. 84, no. 4, pp. 510–513, 1996.

[113] A. Grossmann and J. Morlet, "Decomposition of hardy functions into square integrable wavelets of constant shape," *SIAM Journal on Mathematical Analysis*, vol. 15, no. 4, pp. 723–736, 1984.

[114] L. F. Aguiar and M. J. Soares, "The continuous wavelet transform: A primer," NIPE Working Papers 16/2011, NIPE - Universidade do Minho, 2011.

[115] I. Daubechies, *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992.

[116] Y. Meyer, "Principe d'incertitude, bases hilbertiennes et algèbres d'opérateurs," *Séminaire Bourbaki*, vol. 28, pp. 209–223, 1985-1986.

[117] R. Young, *Wavelet Theory and Its Applications*. Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, 1993.

[118] E. Hall, L. Rouge, and R. Wong, "Hierarchical search for image matching," in *Decision and Control including the 15th Symposium on Adaptive Processes, 1976 IEEE Conference on*, pp. 791–796, Dec 1976.

[119] D. Marr, D. Marr, T. Poggio, and T. Poggio, *A Theory of Human Stereo Vision*. 1977.

[120] A. Rosenfeld and G. Vanderbrug, "Coarse-fine template matching," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 7, pp. 104–107, Feb 1977.

[121] J. J. Koenderink, "The structure of images.," *Biological cybernetics*, vol. 50, no. 5, pp. 363–370, 1984.

[122] W. Grimson, "Computational experiments with a feature based stereo algorithm," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-7, pp. 17–34, Jan 1985.

[123] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, vol. 31, pp. 532–540, Apr. 1983.

[124] J. L. Crowley, "A Representation for Visual Information," *PA Robotics Inst, Carnegie-Mellon University*, 1981.

[125] D. Esteban and C. Galand, "Application of quadrature mirror filters to split band voice coding schemes," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '77.*, vol. 2, pp. 191–195, May 1977.

[126] S. Jaffard, Y. Meyer, and R. Ryan, *Wavelets: Tools for Science and Technology*. Society for Industrial and Applied Mathematics, 2001.

[127] R. González and R. Woods, *Digital Image Processing*. Pearson/Prentice Hall, 2008.

[128] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, vol. 41, no. 7, pp. 909–996, 1988.

[129] R. González, R. Woods, and S. Eddins, *Digital Image Processing Using Matlab*. Pearson Prentice Hall, 2004.

[130] N. Huang and S. Shen, *Hilbert–Huang Transform and Its Applications*. Interdisciplinary Mathematical Sciences, 2014.

[131] a. C. Kak, "Computerized tomography with X-ray, emission, and ultrasound sources," *Proceedings of the IEEE*, vol. 67, no. 9, pp. 1245–1272, 1979.

[132] Mathworks, "iradon function, Inverse Radon Transform." `http://uk.mathworks.com/help/images/ref/iradon.html`.

[133] L. Shepp and B. Logan, "The fourier reconstruction of a head section," *Nuclear Science, IEEE Transactions on*, vol. 21, pp. 21–43, June 1974.

[134] E. P. A. Constantino and K. B. Ozanyan, "Sinogram recovery for sparse angle tomography using a sinusoidal hough transform," *Measurement Science and Technology*, vol. 19, no. 9, p. 094015, 2008.

[135] Mathworks, "Peak Signal-to-Noise Ratio (PSNR)." `http://uk.mathworks.com/help/images/ref/psnr.html`.

[136] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, pp. 600–612, April 2004.

[137] Mathworks, "Structural Similarity Index (SSIM)." `http://uk.mathworks.com/help/images/ref/ssim.html?s_tid=srchtitle`.

[138] A. Boag, Y. Bresler, and E. Michielssen, "A multilevel domain decomposition algorithm for fast o(n2logn) reprojection of tomographic images," *Image Processing, IEEE Transactions on*, vol. 9, no. 9, pp. 1573–1582, 2000.

[139] T. Olson, "Optimal time-frequency projections for localized tomography.," *Annals of biomedical engineering*, vol. 23, no. 5, pp. 622–36, 1995.

[140] A. Bilgot, L. Desbat, and V. Perrier, "FBP and the interior problem in 2D tomography," *IEEE Nuclear Science Symposium Conference Record*, vol. 0, no. 2, pp. 4080–4085, 2012.

[141] P. Tay and J. Havlicek, "Frequency implementation of discrete wavelet transforms," in *Image Analysis and Interpretation, 2004. 6th IEEE Southwest Symposium on*, pp. 167–171, March 2004.

[142] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[143] J. V. Neumann, "First draft of a report on the edvac," tech. rep., 1945.

[144] S. Terepin, "Digital signal processor," Dec. 30 1986. US Patent 4,633,386.

[145] C. Weaver, R. Krishna, L. Wu, and T. Austin, "Application specific architectures: A recipe for fast, flexible and power efficient designs," in *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '01, (New York, NY, USA), pp. 181–185, ACM, 2001.

[146] D. Compston, "Two's Complement Binary Strings." `http://www.mathworks.com/matlabcentral/fileexchange/38889-two-s-complement-binary-strings/all_filesl`.

[147] "Virtex-6 FPGA ML605 Evaluation Kit." `http://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html`.

[148] "ISE Design Suite." `http://www.xilinx.com/products/design-tools/ise-design-suite.html`.

[149] Xilinx, *LogiCORE IP Fast Fourier Transform User's Manual, DS260*.

[150] Xilinx, *LogiCORE IP Complex Multiplier User's Manual, DS291*.

[151] Xilinx, *LogiCORE IP FIR Compiler, DS534*.

[152] S. Bonnet, F. Peyrin, F. Turjman, and R. Prost, "Multiresolution reconstruction in fan-beam tomography," in *Nuclear Science Symposium Conference Record, 2000 IEEE*, vol. 2, pp. 15/105–15/109 vol.2, 2000.

[153] K. Niinimaki, S. Siltanen, and V. Kolehmainen, "Bayesian multiresolution method for local tomography in dental x-ray imaging," *Physics in Medicine and Biology*, vol. 52, no. 22, p. 6663, 2007.

[154] H. Cao, K. Cai, M. Xu, and D. Li, "A new wavelet-based cone-beam local reconstruction algorithm for off-centered circular trajectory," in *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, vol. 1, pp. 56–60, Oct 2010.

[155] S. Oeckl, T. Wenzel, S. Gondrom, and F. Rauch, "Real inline X-ray 3D CT with short cycle times for light metal casting inspection in production," *2008 IEEE Nuclear Science Symposium Conference Record*, pp. 562–564, Oct. 2008.

[156] G. Tisson, P. Scheunders, and D. V. Dyck, "3d region of interest x-ray ct for geometric magnification from multiresolution acquisitions," in *Proc. ISBI04, IEEE International Symposium on Biomedical Imaging, 15-18 april 2004, Arlington, VA*, pp. 567–570, 2004.

[157] "Virtex Ultrascale." `http://www.xilinx.com/products/silicon-devices/fpga/virtex-ultrascale.html`.

Appendix A

MATLAB M-files

A.1 MATLAB *iradon* Function

```
%=========================================================================
%---SINOGRAM CREATION-----------------
%=========================================================================
p = phantom(256);  %Shepp-Logan phantom 256x256
angles = 0:179;    %180 angles from 0 to 179
[sngrm r] = radon(p,angles);   %Sinogram and radial coordinates
%=========================================================================
%---RAMP FILTER DESIGN----------------
%=========================================================================
%bandlimited ramp filter (Eqn. 61 Chapter 3, Kak and Slaney)
len = length(sngrm);
order = max(64,2^nextpow2(len)); %up to the next highest ...
    power of 2
n = 0:order; % 'order' is always even.
filtImpResp = zeros(1,order+1); %the bandlimited ramp's ...
    impulse response vector
filtImpResp(1) = 1/4; % Set the DC term
filtImpResp(2:2:end) = -1./((pi*n(2:2:end)).^2); % Set the ...
    values for odd n (values for even n are 0)
filtImpResp = [filtImpResp filtImpResp(end-1:-1:2)];
%=========================================================================
%---RAMP FILTER IN THE FREQUENCY DOMAIN--------------
%=========================================================================
filt = 2*real(fft(filtImpResp));
filt = filt(1:order+1);
```

```
filt = [filt.' ; filt(end-1:-1:2).'];    % Symmetry of the filter
%=========================================================================
%---FT OF PROJECTION DATA-------------------------------------
%=========================================================================
sngrm(length(filt),1) = 0;  % Zero pad projections
ft_sngrm = fft(sngrm);    % fft of projections
%=========================================================================
%---FILTERING IN FOURIER DOMAIN--------------------------------
%=========================================================================
for i = 1:size(ft_sngrm,2)
    ft_sngrm(:,i) = ft_sngrm(:,i).*filt; % frequency domain ...
       filtering
end
%=========================================================================
%---INVERSE FFT OF FILTERED PROJECTION DATA--------------------
%=========================================================================
proj = real(ifft(ft_sngrm));     %ifft of filtered projections
proj(len+1:end,:) = [];   % Truncate the filtered projections ...
   to its initial size (size of radon transformed data)
%=========================================================================
%---X AND Y CARTESIAN COORDINATES ARRAYS AND MEMORY ALLOCATION--
%=========================================================================
% Define the x & y axes for the reconstructed image so that ...
   the origin
% (center) is in the spot which RADON would choose.
N = 2*floor( size(proj,1)/(2*sqrt(2)) );  % output size ...
   calculated from the projection size
center = floor((N + 1)/2);
xleft = -center + 1;
x = (1:N) - 1 + xleft;  %vector containing -center:center in ...
   the x axis
x = repmat(x, N, 1);    %matrix containing the vectors ...
   mentioned above
ytop = center - 1;
y = (N:-1:1).' - N + ytop;  %vector containing center:-center ...
   in the y axis
y = repmat(y, 1, N);        %matrix containing the vectors ...
   mentioned above
ctrIdx = ceil(len/2);     % index of the center of the ...
   projections
theta = pi*angles/180;  %angles in radians
% Generate trignometric tables
```

```
costheta = cos(theta);
sintheta = sin(theta);
% Allocate memory for the image
img = zeros(N,class(p));
image = zeros(N,class(p));
%=================================================================
%---BACKPROJECTION-------------------------------------
%=================================================================
%For linear interpolation
for i=1:length(theta)
    t_proj = proj(:,i);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    img = img + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx); %linear interpolation
    image = img*(pi/(2*length(theta))); %Normalisation
end
```

A.2 Customised half-frequency spectre, MATLAB *iradon* Function

```
%=================================================================
%---SINOGRAM CREATION----------------------------------------
%=================================================================
p = phantom(256);
angles = 0:179;
[sngrm r] = radon(p,angles);   %Sinogram and radial coordinates
%=================================================================
%---RAMP FILTER DESIGN---------------------------------------
%=================================================================
len = size(sngrm,1);
order = max(64,2^nextpow2(len));
n = 0:order; % 'order' is always even.
filtImpResp = zeros(1,order+1); %the bandlimited ramp's ...
    impulse response
filtImpResp(1) = 1/4; % Set the DC term
filtImpResp(2:2:end) = -1./((pi*n(2:2:end)).^2); % Set the ...
    values for odd n (values for even n are 0)
filtImpResp = [filtImpResp filtImpResp(end-1:-1:2)];
```

```
%=========================================================================
%---RAMP FILTER IN THE FREQUENCY DOMAIN-------------------------
%=========================================================================
filt = (fft(filtImpResp));
filt(order+2:end) = [];  %Truncation of the Filter Frequency ...
   Spectre
%=========================================================================
%---FT OF PROJECTION DATA---------------------------------------
%=========================================================================
sngrm(size(filtImpResp,2),:) = 0;  % Zero pad projections
sngrm = sngrm';
for i = 1:size(sngrm,1)
    ft_sngrm(i,:) = fft(sngrm(i,:));    % fft of projections
end
ft_sngrm = ft_sngrm(:,1:order+1); %Truncation of Projection ...
   Data Frequency Spectre
%=========================================================================
%---FILTERING IN FOURIER DOMAIN---------------------------------
%=========================================================================
for i = 1:length(angles)
    f_projections(i,:) = filt.*ft_sngrm(i,:); % frequency ...
       domain filtering
end
f_projections = [f_projections ...
   conj(f_projections(:,end-1:-1:2))];
%=========================================================================
%---INVERSE FFT OF FILTERED PROJECTION DATA----------------------
%=========================================================================
for i = 1:size(ft_sngrm,1)
    s_projections(i,:) = real(ifft(f_projections(i,:))); ...
       %ifft of filtered projections
end
s_projections(:,(len)+1:end) = []; % Truncate the filtered ...
   projections to its initial size (size of radon transformed ...
   data)
s_projections = s_projections';
%=========================================================================
%---X AND Y CARTESIAN COORDINATES ARRAYS AND MEMORY ALLOCATION---
%=========================================================================
% Define the x & y axes for the reconstructed image
N = size(p,1);
center = floor((N+1)/2);
```

```
xleft = -center + 1;
x = (1:N) - 1 + xleft;  %vector containing -center:center in ...
    the x axis
x = repmat(x, N, 1);    %matrix containing the vectors ...
    mentioned above
ytop = center - 1;
y = (N:-1:1).' - N + ytop;  %vector containing center:-center ...
    in the y axisimg = img*pi/(2*length(theta));
y = repmat(y, 1, N);        %matrix containing the vectors ...
    mentioned above
ctrIdx = ceil(size(s_projections,1)/2); % index of the center ...
    of the projections abs(r(1))+1
theta = pi*angles/180; % Generate trignometric tables
costheta = cos(theta);
sintheta = sin(theta);
img = zeros(N,'like',p);% Allocate memory for the image
image = zeros(N,'like',p);
%=========================================================================
%---BACKPROJECTION-------------------------------------------------
%=========================================================================
for i=1:length(theta)
    t_proj = s_projections(:,i);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    img = img + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);%linear interpolation
    image = img*(pi/(length(theta)));  %Normalisation
end
```

A.3 Wavelet-based FBP, MATLAB Implementation

```
%=========================================================================
%---SINOGRAM CREATION-----------------
%=========================================================================
p = phantom(256);
angles = 0:179;
[sngrm r] = radon(p,angles);
%=========================================================================
```

```
%---RAMP FILTER DESIGN---------------------------------------
%=================================================================
len = size(sngrm,1);
order = max(64,2^nextpow2(len));
n = 0:order; % 'order' is always even.
filtImpResp = zeros(1,order+1);
filtImpResp(1) = 1/4; % Set the DC term
filtImpResp(2:2:end) = -1./((pi*n(2:2:end)).^2);
filtImpResp = [filtImpResp filtImpResp(end-1:-1:2)];
%=================================================================
%---RAMP FILTER IN THE FREQUENCY DOMAIN-----------------------
%=================================================================
filt = (fft(filtImpResp));
filt(order+2:end) = [];
%=================================================================
%---FT OF PROJECTIONS-----------------------------------------
%=================================================================
sngrm(size(filtImpResp,2),:) = 0;
sngrm = sngrm.';
for i = 1:size(sngrm,1)
    ft_sngrm(i,:) = fft(sngrm(i,:));
end
ft_sngrm = ft_sngrm(:,1:order+1);
%=================================================================
%---WAVELET FILTERS-------------------------------------------
%=================================================================
[lp, hp] = wavefilter('db5', 'd'); %load analysis wavelet and ...
   scaling functions
%low-pass filters(scaling functions)
lp_ft1 = lp((length(lp)/2)+1:end);
lp_ft1(order*2) = 0;
lp_ft2 = lp(1:(length(lp)/2));
lp_ft1((end-(length(lp_ft2)))+1:end) = lp_ft2;
lp = lp_ft1; %adjusting of the filters format and zero ...
   padding for FFT
%high-pass filters(wavelet functions)
hp_ft1 = hp((length(hp)/2)+1:end);
hp_ft1(order*2) = 0;
hp_ft2 = hp(1:(length(hp)/2));
hp_ft1((end-(length(hp_ft2)))+1:end) = hp_ft2;
hp = hp_ft1;
%=================================================================
```

```
%---FT OF WAVELET FILTERS-------------------------------------
%==================================================================
f_lp = fftshift(fft(lp));
f_hp = fftshift(fft(hp));
%==================================================================
%---CARTESIAN TO POLAR-----------------------------------------
%==================================================================
theta = pi*angles/180; %generate trigonometric tables
costheta = cos(theta);
sintheta = sin(theta);
w = -order:order-1;
for i=1:length(costheta)
    uu(i,:) = costheta(i).*w;
    vv(i,:) = sintheta(i).*w;
end
u = floor(uu)+order+1;
v = floor(vv)+order+1;
nl = f_lp(u); %low-pas along rows
nh = f_hp(u); %high-pas along rows
ml = f_lp(v); %high-pas along columns
mh = f_hp(v); %high-pas along columns
mh(:,1:order-1) = []; %truncate Fourier domain filters to ...
ml(:,1:order-1) = []; %consider only positive frequencies
nh(:,1:order-1) = [];
nl(:,1:order-1) = [];
mh = conj(mh);
ml = conj(ml);
%==================================================================
%---WAVELET RAMP FILTERING IN FOURIER DOMAIN-------------------
%==================================================================
for i = 1:length(angles)
    app_ramp(i,:) = filt.*ml(i,:).*nl(i,:);
    hor_ramp(i,:) = filt.*ml(i,:).*nh(i,:);
    vert_ramp(i,:) = filt.*mh(i,:).*nl(i,:);
    diag_ramp(i,:) = filt.*mh(i,:).*nh(i,:);
end
%=========================wavelet filtering=====================
for i = 1:length(angles)
    hor_filt(i,:) = ft_sngrm(i,:).*hor_ramp(i,:);
    vert_filt(i,:) = ft_sngrm(i,:).*vert_ramp(i,:);
    diag_filt(i,:) = ft_sngrm(i,:).*diag_ramp(i,:);
    app_filt(i,:) = ft_sngrm(i,:).*app_ramp(i,:);
```

```
end
%==========================negative frequencies================
hor_filt = [hor_filt conj(hor_filt(:,end-1:-1:2))];
vert_filt = [vert_filt conj(vert_filt(:,end-1:-1:2))];
diag_filt = [diag_filt conj(diag_filt(:,end-1:-1:2))];
app_filt = [app_filt conj(app_filt(:,end-1:-1:2))];
%====================ifft=======================================
for i = 1:size(ft_sngrm,1)
    w_diag(i,:) = real(ifft(diag_filt(i,:)));
    w_vert(i,:) = real(ifft(vert_filt(i,:)));
    w_hor(i,:) = real(ifft(hor_filt(i,:)));
    w_app(i,:) = real(ifft(app_filt(i,:)));
end
%============================removal of added data for fft=====
w_diag(:,(len)+1:end) = [];
w_vert(:,(len)+1:end) = [];
w_hor(:,(len)+1:end) = [];
w_app(:,(len)+1:end) = [];
%================================================================
%---X AND Y CARTESIAN COORDINATES ARRAYS AND MEMORY ALLOCATION--
%================================================================
N = size(p,1);
center = floor((N+1)/2);
resolution = 2; %subsampling factor
xleft = -center + 1;
x = (1:resolution:N) - 1 + xleft;
x = repmat(x, N/resolution, 1);
ytop = center - 1;
y = (N:-resolution:1).' - N + ytop;
y = repmat(y, 1, N/resolution);
ctrIdx = ceil(len/2);
t_proj = zeros(1,size(w_app,2));
%Memory allocation for coefficient images
imgd = zeros((N)/resolution,class(p));
imgv = zeros((N)/resolution,class(p));
imgh = zeros((N)/resolution,class(p));
imga = zeros((N)/resolution,class(p));
%================================================================
%---BACKPROJECTION-----------------------------------------------
%================================================================
%--------------approximations------------
for i=1:length(theta) %backprojection involving all angles
```

```
132     t proj = w app(i,:);
133     t = x*costheta(i) + y*sintheta(i);
134     a = floor(t);
135     imga = imga + (t-a).* t proj(a+1+ctrIdx) + ...
136        (a+1-t).* t proj(a+ctrIdx);
137     imagea = imga*(pi/(length(theta)));
138 end
139 %------------------horizontal--------------
140 for i=13:169 %backprojection involving angles in the range from
141     t proj = w hor(i,:); %13 to 169, of 180
142     t = x*costheta(i) + y*sintheta(i);
143     a = floor(t);
144     imgh = imgh + (t-a).* t proj(a+1+ctrIdx) + ...
145        (a+1-t).* t proj(a+ctrIdx);
146     imageh = imgh*(pi/(length(theta)));
147 end
148 %--------------vertical--------------------
149 for i=1:79 %1 to 9
150     t proj = w vert(i,:);
151     t = x*costheta(i) + y*sintheta(i);
152     a = floor(t);
153     imgv = imgv + (t-a).* t proj(a+1+ctrIdx)
154        (a+1-t).* t proj(a+ctrIdx);
155     imagev = imgv*(pi/(length(theta)));
156 end
157 for i=103:180 %103 to 180
158     t proj = w vert(i,:);
159     t = x*costheta(i) + y*sintheta(i);
160     a = floor(t);
161     imgv = imgv + (t-a).* t proj(a+1+ctrIdx)
162        (a+1-t).* t proj(a+ctrIdx);
163     imagev = imgv*(pi/(length(theta)));
164 end
165 %--------------diagonal--------------------
166 for i=8:83 %8 to 83
167     t proj = w diag(i,:);
168     t = x*costheta(i) + y*sintheta(i);
169     a = floor(t);
170     imgd = imgd + (t-a).* t proj(a+1+ctrIdx)
171        (a+1-t).* t proj(a+ctrIdx);
172     imaged = imgd*(pi/(length(theta)));
173 end
```

```
for i=99:174 %99 to 174
    t_proj = w_diag(i,:);
    t = x*costheta(i) + y*sintheta(i);
    a = floor(t);
    imgd = imgd + (t-a).* t_proj(a+1+ctrIdx)
       (a+1-t).* t_proj(a+ctrIdx);
    imaged = imgd*(pi/(length(theta)));
end
%=======================================================================
%---WAVELET SYNTHESIS-----------------------------------
%=======================================================================
[lp, hp] = wavefilter('db5', 'r'); %Inverse filters
fl = length(lp);
rdiag = upsample(imaged, 2); %coefficient images upsampling
rvert = upsample(imagev, 2); %along rows by a factor of two
rhor = upsample(imageh, 2);
rapp = upsample(imagea, 2);
rdiag = conv2(rdiag, hp.', 'same'); %reverse filtering
rvert = conv2(rvert, lp.', 'same'); %along columns
higher = rdiag + rvert;
rhor = conv2(rhor, hp.', 'same');
rapp = conv2(rapp, lp.', 'same');
lower = rhor + rapp;
higher = higher.';
lower = lower.';
higher = upsample(higher, 2); %upsampling along columns
lower = upsample(lower, 2);
higher = higher.';
lower = lower.';
t_lower = conv2(lower, lp, 'same'); %reverse filtering along
t_higher = conv2(higher, hp, 'same'); %rows
image = (t_higher + t_lower); %output full resolution image
```

A.4 Wavelet-based, Parallel Multiresoltuion FBP, MATLAB Implementation

```
%=======================================================================
%---SINOGRAM CREATION-----------------
%=======================================================================
```

```
p = phantom(256);
nrm = norm(p,2);
angles = 0:179;
sngrm = radon(p,angles);   %Sinogram and radial coordinates
theta = pi*angles/180;
costheta = cos(theta);
sintheta = sin(theta);
%=========================================================================
%---RAMP FILTER DESIGN-----------------------------------------
%=========================================================================
len = size(sngrm,1);
order = max(64,2^nextpow2(len));
n = 0:order;
filtImpResp = zeros(1,order+1);
filtImpResp(1) = 1/4;
filtImpResp(2:2:end) = -1./((pi*n(2:2:end)).^2);
filtImpResp = [filtImpResp filtImpResp(end-1:-1:2)];
%=========================================================================
%---RAMP FILTER IN THE FREQUENCY DOMAIN------------------------
%=========================================================================
filt = (fft(filtImpResp));
filt(order+2:end) = [];
filt = repmat(filt, length(angles), 1);
%=========================================================================
%---FT OF PROJECTION DATA--------------------------------------
%=========================================================================
sngrm(size(filtImpResp,2),:) = 0;  % Zero pad projections
sngrm = sngrm.';
for i = 1:size(sngrm,1)
    ft_sngrm(i,:) = fft(sngrm(i,:));    % fft of projections
end
ft_sngrm = ft_sngrm(:,1:order+1);
%=========================================================================
%---WAVELET FILTERS--------------------------------------------
%=========================================================================
[lp, hp] = wavefilter('bior2.4', 'd');
lp_ft1 = lp((length(lp)/2)+1:end);
lp_ft1(order*2) = 0;
lp_ft2 = lp(1:(length(lp)/2));
lp_ft1((end-(length(lp_ft2)))+1:end) = lp_ft2;
lp = lp_ft1;
hp_ft1 = hp((length(hp)/2)+1:end);
```

```
hp_ft1(order*2) = 0;
hp_ft2 = hp(1:(length(hp)/2));
hp_ft1((end-(length(hp_ft2)))+1:end) = hp_ft2;
hp = hp_ft1;
f_lp = (fft(lp));
f_hp = (fft(hp));
%L2 Filters calculation
f_lp_2 = zeros(1,length(f_lp));
f_hp_2 = zeros(1,length(f_lp));
f_lp_m = [f_lp f_lp];
f_hp_m = [f_hp f_hp];
for i =1:length(f_lp)
    f_lp_2(i) = (f_lp(i).*f_lp_m(2*i));
    f_hp_2(i) = (f_hp_m(2*i).*f_lp(i));
end
%L3 Filters calculation
f_lp_3 = zeros(1,size(f_lp,2));
f_hp_3 = zeros(1,size(f_lp,2));
f_lp_m_3 = [f_lp_m f_lp_m];
f_hp_m_3 = [f_hp_m f_hp_m];
for i =1:length(f_lp)
    f_lp_3(i) = (f_lp_2(i).*f_lp_m_3(4*i));
    f_hp_3(i) = (f_hp_m_3(4*i).*f_lp_2(i));
end
%=========================================================================
%Synthesis Filters
%=========================================================================
[ls, hs] = wavefilter('bior2.4', 'r');
ln_t = length(ls);
ls_ft1 = ls((length(ls)/2)+1:end);
ls_ft1(order*2) = 0;
ls_ft2 = ls(1:(length(ls)/2));
ls_ft1((end-(length(ls_ft2)))+1:end) = ls_ft2;
ls = ls_ft1;
hs_ft1 = hs((length(hs)/2)+1:end);
hs_ft1(order*2) = 0;
hs_ft2 = hs(1:(length(hs)/2));
hs_ft1((end-(length(hs_ft2)))+1:end) = hs_ft2;
hs = hs_ft1;
f_ls = (fft(ls));
f_hs = (fft(hs));
%L2 iFilters calculation
```

```
f_ls_2 = zeros(1,length(f_ls));
f_hs_2 = zeros(1,length(f_ls));
f_ls_m = [f_ls f_ls];
f_hs_m = [f_hs f_hs];
for i =1:length(f_ls)
    f_ls_2(i) = (f_ls(i).*f_ls_m(2*i));
    f_hs_2(i) = (f_hs_m(2*i).*f_ls(i));
end
%L3 iFilters calculation
f_ls_3 = zeros(1,size(f_ls,2));
f_hs_3 = zeros(1,size(f_ls,2));
f_ls_m_3 = [f_ls_m f_ls_m];
f_hs_m_3 = [f_hs_m f_hs_m];
for i =1:length(f_ls)
    f_ls_3(i) = (f_ls_2(i).*f_ls_m_3(4*i));
    f_hs_3(i) = (f_hs_m_3(4*i).*f_ls_2(i));
end
t_ls = real(ifft(f_ls,'symmetric'));
t_ls_2 = real(ifft(f_ls_2, 'symmetric'));
t_ls_3 = real(ifft(f_ls_3, 'symmetric'));
t_hs = real(ifft(f_hs, 'symmetric'));
t_hs_2 = real(ifft(f_hs_2, 'symmetric'));
t_hs_3 = real(ifft(f_hs_3, 'symmetric'));
ln = length(t_ls);
t_ls = [t_ls((end-ln/2)+1:end) t_ls(1:ln/2)];
t_hs = [t_hs((end-ln/2)+1:end) t_hs(1:ln/2)];
t_ls_2 = [t_ls_2((end-ln/2):end) t_ls_2(1:(ln/2)+1)];
t_hs_2 = [t_hs_2((end-ln/2):end) t_hs_2(1:(ln/2)+1)];
t_ls_3 = [t_ls_3((end-ln/2)-2:end) t_ls_3(1:(ln/2)+3)];
t_hs_3 = [t_hs_3((end-ln/2)-2:end) t_hs_3(1:(ln/2)+3)];
t_ls = t_ls((ln/2-ln_t/2)+1:ln/2+ln_t/2);
t_hs = t_hs((ln/2-ln_t/2)+1:ln/2+ln_t/2);
t_ls_2 = t_ls_2((ln/2-ln_t)+1:ln/2+ln_t);
t_hs_2 = t_hs_2((ln/2-ln_t)+1:ln/2+ln_t);
t_ls_3 = t_ls_3((ln/2-ln_t*2)+1:ln/2+ln_t*2);
t_hs_3 = t_hs_3((ln/2-ln_t*2)+1:ln/2+ln_t*2);
f_lp = (fftshift(f_lp));
f_hp = (fftshift(f_hp));
f_lp_2 = (fftshift(f_lp_2));
f_hp_2 = (fftshift(f_hp_2));
f_lp_3 = (fftshift(f_lp_3));
f_hp_3 = (fftshift(f_hp_3));
```

```
%=========================================================================
%---WAVELET RAMP FILTERS-----------------------------------------
%=========================================================================
w = -order:order-1;
for i=1:length(angles)
    uu(i,:) = costheta(i).*w;
    vv(i,:) = sintheta(i).*w;
end
u = floor(uu)+order+1;
v = floor(vv)+order+1;
%L1 Filters
ml = f_lp(u);
mh = f_hp(u);
nl = f_lp(v);
nh = f_hp(v);%
ml(:,1:order-1) = [];
mh(:,1:order-1) = [];
nl(:,1:order-1) = [];
nh(:,1:order-1) = [];
nl = conj(nl);
nh = conj(nh);
%L2 Filters
ml_2 = f_lp_2(u);
mh_2 = f_hp_2(u);
nl_2 = f_lp_2(v);
nh_2 = f_hp_2(v);%
ml_2(:,1:order-1) = [];
mh_2(:,1:order-1) = [];
nl_2(:,1:order-1) = [];
nh_2(:,1:order-1) = [];
nl_2 = conj(nl_2);
nh_2 = conj(nh_2);
%L3 Filters
ml_3 = f_lp_3(u);
mh_3 = f_hp_3(u);
nl_3 = f_lp_3(v);
nh_3 = f_hp_3(v);%
ml_3(:,1:order-1) = [];
mh_3(:,1:order-1) = [];
nl_3(:,1:order-1) = [];
nh_3(:,1:order-1) = [];
nl_3 = conj(nl_3);
```

```
172 nh_3 = conj(nh_3);
173 hor_filt = ml.*nh.*filt;
174 vert_filt = mh.*nl.*filt;
175 diag_filt = mh.*nh.*filt;
176 hor_filt_2 = ml_2.*nh_2.*filt;
177 vert_filt_2 = mh_2.*nl_2.*filt;
178 diag_filt_2 = mh_2.*nh_2.*filt;
179 hor_filt_3 = ml_3.*nh_3.*filt;
180 vert_filt_3 = mh_3.*nl_3.*filt;
181 diag_filt_3 = mh_3.*nh_3.*filt;
182 app_filt_3 = ml_3.*nl_3.*filt;
183 %================================================================
184 %---FILTERING-----------------------------------------------------
185 %================================================================
186 hor_filt = hor_filt.*ft_sngrm;
187 vert_filt = vert_filt.*ft_sngrm;
188 diag_filt = diag_filt.*ft_sngrm;
189 hor_filt_2 = hor_filt_2.*ft_sngrm;
190 vert_filt_2 = vert_filt_2.*ft_sngrm;
191 diag_filt_2 = diag_filt_2.*ft_sngrm;
192 app_filt_3 = app_filt_3.*ft_sngrm;
193 hor_filt_3 = hor_filt_3.*ft_sngrm;
194 vert_filt_3 = vert_filt_3.*ft_sngrm;
195 diag_filt_3 = diag_filt_3.*ft_sngrm;
196 %==========================negative frequencies================
197 hor_filt = [hor_filt conj(hor_filt(:,end-1:-1:2))];
198 vert_filt = [vert_filt conj(vert_filt(:,end-1:-1:2))];
199 diag_filt = [diag_filt conj(diag_filt(:,end-1:-1:2))];
200 hor_filt_2 = [hor_filt_2 conj(hor_filt_2(:,end-1:-1:2))];
201 vert_filt_2 = [vert_filt_2 conj(vert_filt_2(:,end-1:-1:2))];
202 diag_filt_2 = [diag_filt_2 conj(diag_filt_2(:,end-1:-1:2))];
203 hor_filt_3 = [hor_filt_3 conj(hor_filt_3(:,end-1:-1:2))];
204 vert_filt_3 = [vert_filt_3 conj(vert_filt_3(:,end-1:-1:2))];
205 diag_filt_3 = [diag_filt_3 conj(diag_filt_3(:,end-1:-1:2))];
206 app_filt_3 = [app_filt_3 conj(app_filt_3(:,end-1:-1:2))];
207 %====================ifft=======================================
208     for i = 1:length(angles)
209         w_diag(i,:) = real(ifft(diag_filt(i,:)));
210         w_vert(i,:) = real(ifft(vert_filt(i,:)));
211         w_hor(i,:) = real(ifft(hor_filt(i,:)));
212         w_diag_2(i,:) = real(ifft(diag_filt_2(i,:)));
213         w_vert_2(i,:) = real(ifft(vert_filt_2(i,:)));
```

```
214         w_hor_2(i,:) = real(ifft(hor_filt_2(i,:)));
215         w_diag_3(i,:) = real(ifft(diag_filt_3(i,:)));
216         w_vert_3(i,:) = real(ifft(vert_filt_3(i,:)));
217         w_hor_3(i,:) = real(ifft(hor_filt_3(i,:)));
218         w_app_3(i,:) = real(ifft(app_filt_3(i,:)));
219     end
220 %==============================removal of added data for fft=====
221 w_diag(:,(len)+1:end) = [];
222 w_vert(:,(len)+1:end) = [];
223 w_hor(:,(len)+1:end) = [];
224 w_diag_2(:,(len)+1:end) = [];
225 w_vert_2(:,(len)+1:end) = [];
226 w_hor_2(:,(len)+1:end) = [];
227 w_diag_3(:,(len)+1:end) = [];
228 w_vert_3(:,(len)+1:end) = [];
229 w_hor_3(:,(len)+1:end) = [];
230 w_app_3(:,(len)+1:end) = [];
231 w_diag = w_diag.';
232 w_vert = w_vert.';
233 w_hor = w_hor.';
234 w_diag_2 = w_diag_2.';
235 w_vert_2 = w_vert_2.';
236 w_hor_2 = w_hor_2.';
237 w_diag_3 = w_diag_3.';
238 w_vert_3 = w_vert_3.';
239 w_hor_3 = w_hor_3.';
240 w_app_3 = w_app_3.';
241 %=================================================================
242 %---BACKPROJECTION L1-----------------------------------------
243 %=================================================================
244 N = size(p,1);
245 center = floor((N+1)/2);
246 resolution = 2;
247 xleft = -center + 1;
248 x = (1:resolution:N) - 1 + xleft;
249 x = repmat(x, ceil(N/resolution), 1);
250 ytop = center - 1;
251 y = (N:-resolution:1).' - N + ytop;
252 y = repmat(y, 1, ceil(N/resolution));
253
254 ctrIdx = ceil(len/2);
255 imgd = zeros(ceil((N)/resolution),class(p));
```

```
256 imgv = zeros(ceil((N)/resolution),class(p));
257 imgh = zeros(ceil((N)/resolution),class(p));
258 %=================================================================
259 %--------------diagonal-----------------------
260 for i=7:84%1:length(angles)%26:64
261     t_proj = w_diag(:,i);
262     t = x*costheta(i) + y*sintheta(i);
263     a = floor(t);
264     imgd = imgd + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
265     imgda = imgd*(pi/(length(theta)));
266 end
267 for i=98:178%1:length(angles)%26:64
268     t_proj = w_diag(:,i);
269     t = x*costheta(i) + y*sintheta(i);
270     a = floor(t);
271     imgd = imgd + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
272     imgda = imgd*(pi/(length(theta)));
273 end
274 %--------------vertical--------------------
275 for i=1:85%1:length(angles)%1:45
276     t_proj = w_vert(:,i);
277     t = x*costheta(i) + y*sintheta(i);
278     a = floor(t);
279     imgv = imgv + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
280     imgva = imgv*(pi/(length(theta)));
281 end
282 for i=97:180%1:length(angles)%1:45
283     t_proj = w_vert(:,i);
284     t = x*costheta(i) + y*sintheta(i);
285     a = floor(t);
286     imgv = imgv + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
287     imgva = imgv*(pi/(length(theta)));
288 end
289 %-----------------horizontal-----------------------------------
290 for i=6:176%1:length(angles)%45:135
291     t_proj = w_hor(:,i);
292     t = x*costheta(i) + y*sintheta(i);
293     a = floor(t);
```

```
294     imgh = imgh + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
295     imgha = imgh*(pi/(length(theta)));
296 end
297 %=========================================================================
298 %---BACKPROJECTION L2-----------------------------------------
299 %=========================================================================
300 resolution = 4;
301 x_2 = (1:resolution:N) - 1 + xleft;
302 x_2 = repmat(x_2, (N/resolution), 1);
303 y_2 = (N:-resolution:1).' - N + ytop;
304 y_2 = repmat(y_2, 1, (N/resolution));
305 t_2 = zeros(size(x_2));
306 a_2 = zeros(size(x_2));
307 imgd_2 = zeros(((N)/resolution),class(p));
308 imgv_2 = zeros(((N)/resolution),class(p));
309 imgh_2 = zeros(((N)/resolution),class(p));
310 %-------------diagonal-----------------------
311 for i=4:87%1:length(angles)%26:64%length(theta)
312     t_proj = w_diag_2(:,i);
313     t_2 = x_2*costheta(i) + y_2*sintheta(i);
314     a_2 = floor(t_2);
315     imgd_2 = imgd_2 + (t_2-a_2).*t_proj(a_2+1+ctrIdx) + ...
            (a_2+1-t_2).*t_proj(a_2+ctrIdx);
316     imgda_2 = imgd_2*((pi/length(theta)));
317 end
318 for i=95:177%1:length(angles)%26:64%length(theta)
319     t_proj = w_diag_2(:,i);
320     t_2 = x_2*costheta(i) + y_2*sintheta(i);
321     a_2 = floor(t_2);
322     imgd_2 = imgd_2 + (t_2-a_2).*t_proj(a_2+1+ctrIdx) + ...
            (a_2+1-t_2).*t_proj(a_2+ctrIdx);
323     imgda_2 = imgd_2*((pi/length(theta)));
324 end
325 %-------------vertical--------------------
326 for i=1:86%1:length(angles)%1:45
327     t_proj = w_vert_2(:,i);
328     t_2 = x_2*costheta(i) + y_2*sintheta(i);
329     a_2 = floor(t_2);
330     imgv_2 = imgv_2 + (t_2-a_2).*t_proj(a_2+1+ctrIdx) + ...
            (a_2+1-t_2).*t_proj(a_2+ctrIdx);
331     imgva_2 = imgv_2*((pi/length(theta)));
```

```
end
for i=96:180%1:length(angles)%1:45
    t_proj = w_vert_2(:,i);
    t_2 = x_2*costheta(i) + y_2*sintheta(i);
    a_2 = floor(t_2);
    imgv_2 = imgv_2 + (t_2-a_2).*t_proj(a_2+1+ctrIdx) + ...
        (a_2+1-t_2).*t_proj(a_2+ctrIdx);
    imgva_2 = imgv_2*((pi/length(theta)));
end
%-----------------horizontal-----------------------------------
for i=5:175%1:length(angles)%45:135
    t_proj = w_hor_2(:,i);
    t_2 = x_2*costheta(i) + y_2*sintheta(i);
    a_2 = floor(t_2);

    %imgh_2 = imgh_2 + t_proj(a_2+1+ctrIdx);
    imgh_2 = imgh_2 + (t_2-a_2).*t_proj(a_2+1+ctrIdx) + ...
        (a_2+1-t_2).*t_proj(a_2+ctrIdx);
    imgha_2 = imgh_2*((pi/length(theta)));
end
%=================================================================
%---BACKPROJECTION L3-------------------------------------------
%=================================================================
resolution = 8;
x_3 = (1:resolution:N) - 1 + xleft;
x_3 = repmat(x_3, (N/resolution), 1);
y_3 = (N:-resolution:1).' - N + ytop;
y_3 = repmat(y_3, 1, (N/resolution));
imgd_3 = zeros(((N)/resolution),class(p));
imgv_3 = zeros(((N)/resolution),class(p));
imgh_3 = zeros(((N)/resolution),class(p));
imga_3 = zeros(((N)/resolution),class(p));
%-------------diagonal-----------------------
for i=3:88%1:length(angles)%26:64
    t_proj = w_diag_3(:,i);
    t_3 = x_3*costheta(i) + y_3*sintheta(i);
    a_3 = floor(t_3);
    imgd_3 = imgd_3 + (t_3-a_3).*t_proj(a_3+1+ctrIdx) + ...
        (a_3+1-t_3).*t_proj(a_3+ctrIdx);
    imgda_3 = imgd_3*(pi/(length(theta)));
end
for i=94:179%1:length(angles)%26:64
```

```
371     t_proj = w_diag_3(:,i);
372     t_3 = x_3*costheta(i) + y_3*sintheta(i);
373     a_3 = floor(t_3);
374     imgd_3 = imgd_3 + (t_3-a_3).*t_proj(a_3+1+ctrIdx) + ...
            (a_3+1-t_3).*t_proj(a_3+ctrIdx);
375     imgda_3 = imgd_3*(pi/(length(theta)));
376 end
377 %--------------vertical---------------------
378 for i=1:87%length(angles)%1:45
379     t_proj = w_vert_3(:,i);
380     t_3 = x_3*costheta(i) + y_3*sintheta(i);
381     a_3 = floor(t_3);
382     imgv_3 = imgv_3 + (t_3-a_3).*t_proj(a_3+1+ctrIdx) + ...
            (a_3+1-t_3).*t_proj(a_3+ctrIdx);
383     imgva_3 = imgv_3*(pi/(length(theta)));
384 end
385 for i=95:180%length(angles)%1:45
386     t_proj = w_vert_3(:,i);
387     t_3 = x_3*costheta(i) + y_3*sintheta(i);
388     a_3 = floor(t_3);
389     imgv_3 = imgv_3 + (t_3-a_3).*t_proj(a_3+1+ctrIdx) + ...
            (a_3+1-t_3).*t_proj(a_3+ctrIdx);
390     imgva_3 = imgv_3*(pi/(length(theta)));
391 end
392 %-----------------horizontal-----------------------------------
393 for i=4:178%length(angles)%45:135
394     t_proj = w_hor_3(:,i);
395     t_3 = x_3*costheta(i) + y_3*sintheta(i);
396     a_3 = floor(t_3);
397     imgh_3 = imgh_3 + (t_3-a_3).*t_proj(a_3+1+ctrIdx) + ...
            (a_3+1-t_3).*t_proj(a_3+ctrIdx);
398     imgha_3 = imgh_3*(pi/(length(theta)));
399 end
400 %---------------------approximations---------------------------
401 for i=1:length(angles)
402     t_proj = w_app_3(:,i);
403     t_3 = x_3*costheta(i) + y_3*sintheta(i);
404     a_3 = floor(t_3);
405     imga_3 = imga_3 + (t_3-a_3).*t_proj(a_3+1+ctrIdx) + ...
            (a_3+1-t_3).*t_proj(a_3+ctrIdx);
406     imgaa_3 = imga_3*(pi/(length(theta)));
407 end
```

```
%=================================================================
%---WAVELET RECONSTRUCTION L3---------------------------------
%=================================================================
lp = t_ls_3;
hp = t_hs_3;
rdiag_3 = upsample((imgda_3), 8, 3);
rvert_3 = upsample((imgva_3), 8, 3);
rhor_3 = upsample((imgha_3), 8, 3);
rapp_3 = upsample(imgaa_3, 8, 3);
rdiag_3 = rdiag_3.';
rvert_3 = rvert_3.';
rhor_3 = rhor_3.';
rapp_3 = rapp_3.';
rdiag_3 = upsample(rdiag_3, 8, 3);
rvert_3 = upsample(rvert_3, 8, 3);
rhor_3 = upsample(rhor_3, 8, 3);
rapp_3 = upsample(rapp_3, 8, 3);
rdiag_3 = rdiag_3.';
rvert_3 = rvert_3.';
rhor_3 = rhor_3.';
rapp_3 = rapp_3.';
rdiag_3 = conv2(rdiag_3, hp.', 'same');
rvert_3 = conv2(rvert_3, lp.', 'same');
rhor_3 = conv2(rhor_3, hp.', 'same');
rapp_3 = conv2(rapp_3, lp.', 'same');
rdiag_3 = conv2(rdiag_3, hp, 'same');
rvert_3 = conv2(rvert_3, hp, 'same');
rhor_3 = conv2(rhor_3, lp, 'same');
rapp_3 = conv2(rapp_3, lp, 'same');
image_3 = (rapp_3 + rdiag_3 + rvert_3 + rhor_3);
%=================================================================
%---WAVELET RECONSTRUCTION L2---------------------------------
%=================================================================
lp = t_ls_2;
hp = t_hs_2;
%Upsampling-------------------------
rdiag_2 = upsample((imgda_2), 4, 1);
rvert_2 = upsample((imgva_2), 4, 1);
rhor_2 = upsample((imgha_2), 4, 1);
rdiag_2 = rdiag_2.';
rvert_2 = rvert_2.';
rhor_2 = rhor_2.';
```

```
rdiag_2 = upsample(rdiag_2, 4, 1);
rvert_2 = upsample(rvert_2, 4, 1);
rhor_2 = upsample(rhor_2, 4, 1);
rdiag_2 = rdiag_2.';
rvert_2 = rvert_2.';
rhor_2 = rhor_2.';
rdiag_2 = conv2(rdiag_2, hp.', 'same');
rvert_2 = conv2(rvert_2, lp.', 'same');
rhor_2 = conv2(rhor_2, hp.', 'same');
rdiag_2 = conv2(rdiag_2, hp, 'same');
rvert_2 = conv2(rvert_2, hp, 'same');
rhor_2 = conv2(rhor_2, lp, 'same');
image_2 = (rdiag_2 + rvert_2 + rhor_2);
%=====================================================================
%---WAVELET RECONSTRUCTION L1-----------------------------------
%=====================================================================
lp = t_ls;
hp = t_hs;
%Upsampling---------------------------
rdiag = upsample((imgda), 2);
rvert = upsample((imgva), 2);
rhor = upsample((imgha), 2);
rdiag = rdiag.';
rvert = rvert.';
rhor = rhor.';
rdiag = upsample(rdiag, 2);
rvert = upsample(rvert, 2);
rhor = upsample(rhor, 2);
rdiag = rdiag.';
rvert = rvert.';
rhor = rhor.';
rdiag = conv2(rdiag, hp.', 'same');
rvert = conv2(rvert, lp.', 'same');
rhor = conv2(rhor, hp.', 'same');
rdiag = conv2(rdiag, hp, 'same');
rvert = conv2(rvert, hp, 'same');
rhor = conv2(rhor, lp, 'same');
image = (rdiag + rvert + rhor);
imagen = (image + image_2 + image_3); %output image
```

A.5 Wavelet-based, Multiresolution Parallel Block FBP, MATLAB Implementation

```
%=========================================================================
%---SINOGRAM CREATION-----------------
%=========================================================================
l = 16;   %Parametters to define block distribution
l_2 = 64; %within coefficient images at scales l
p = phantom(256);
angles = 0:179;
[sngrm r] = radon(p,angles);   %Sinogram and radial coordinates
theta = pi*angles/180;
costheta = cos(theta);
sintheta = sin(theta);
len_sngrm = size(sngrm,1);
len = size(sngrm,1);
sngrm_templates(256); %function to generate templates for ...
   block truncation
%=========================================================================
%---REGIONS OF INTEREST---------------------------------------
%=========================================================================
for i = 1:16    %load block sinograms
    filename = sprintf('templates/template_2_%d.png',i);
    grid_template{i} = (imread(filename));
end
for i = 1:4     %block sinograms
    filename = sprintf('templates/template_3_%d.png',i);
    grid_template_2{i} = (imread(filename));
end
for j = 1:length(grid_template) %data truncation for l=1
    for i=1:size(sngrm,2)
        sngrm_roi{j}(i,:) = sngrm(:,i).*grid_template{j}(:,i);
    end
end
for j = 1:length(grid_template_2) %data truncation for l=2
    for i=1:size(sngrm,2)
        sngrm_roi_2{j}(i,:) = ...
            sngrm(:,i).*grid_template_2{j}(:,i);
    end
end
```

```
sngrm = sngrm.';
sngrm_roi_3 = sngrm; %lower resolution scale, not truncated
%=========================================================================
%---RAMP FILTER DESIGN---------------------------------------
%=========================================================================
order = (max(64,2^nextpow2(len)));
filt = (ramp_filt(order));
%=========================================================================
%---WAVELET FILTERS------------------------------------------
%=========================================================================
[lp, hp] = wavefilter('bior2.4', 'd');
[f_lp,f_hp] = (ft_wavefilter(lp, hp, order));
%L2 Filters calculation
f_lp_2 = zeros(1,length(f_lp));
f_hp_2 = zeros(1,length(f_lp));
f_lp_m = [f_lp f_lp];
f_hp_m = [f_hp f_hp];
for i =1:length(f_lp)
    f_lp_2(i) = (f_lp(i).*f_lp_m(2*i));
    f_hp_2(i) = (f_hp_m(2*i).*f_lp(i));
end
%L3 Filters calculation
f_lp_3 = zeros(1,size(f_lp,2));
f_hp_3 = zeros(1,size(f_lp,2));
f_lp_m_3 = [f_lp_m f_lp_m];
f_hp_m_3 = [f_hp_m f_hp_m];
for i =1:length(f_lp)
    f_lp_3(i) = (f_lp_2(i).*f_lp_m_3(4*i));
    f_hp_3(i) = (f_hp_m_3(4*i).*f_lp_2(i));
end
%=========================================================================
%Synthesis Filters
%=========================================================================
[ls, hs] = wavefilter('bior2.4', 'r');
ln_t = length(ls);
[f_ls, f_hs] = ft_wavefilter(ls, hs, order);
%L2 iFilters calculation
f_ls_2 = zeros(1,length(f_ls));
f_hs_2 = zeros(1,length(f_ls));
f_ls_m = [f_ls f_ls];
f_hs_m = [f_hs f_hs];
for i =1:length(f_ls)
```

```
78      f_ls_2(i) = (f_ls(i).*f_ls_m(2*i));
79      f_hs_2(i) = (f_hs_m(2*i).*f_ls(i));
80  end
81  %L3 iFilters calculation
82  f_ls_3 = zeros(1,size(f_ls,2));
83  f_hs_3 = zeros(1,size(f_ls,2));
84  f_ls_m_3 = [f_ls_m f_ls_m];
85  f_hs_m_3 = [f_hs_m f_hs_m];
86  for i =1:length(f_ls)
87      f_ls_3(i) = (f_ls_2(i).*f_ls_m_3(4*i));
88      f_hs_3(i) = (f_hs_m_3(4*i).*f_ls_2(i));
89  end
90  t_ls = real(ifft(f_ls,'symmetric'));
91  t_ls_2 = real(ifft(f_ls_2, 'symmetric'));
92  t_ls_3 = real(ifft(f_ls_3, 'symmetric'));
93  t_hs = real(ifft(f_hs, 'symmetric'));
94  t_hs_2 = real(ifft(f_hs_2, 'symmetric'));
95  t_hs_3 = real(ifft(f_hs_3, 'symmetric'));
96  ln = length(t_ls);
97  t_ls = [t_ls((end-ln/2)+1:end) t_ls(1:ln/2)];
98  t_hs = [t_hs((end-ln/2)+1:end) t_hs(1:ln/2)];
99  t_ls_2 = [t_ls_2((end-ln/2):end) t_ls_2(1:(ln/2)+1)];
100 t_hs_2 = [t_hs_2((end-ln/2):end) t_hs_2(1:(ln/2)+1)];
101 t_ls_3 = [t_ls_3((end-ln/2)-2:end) t_ls_3(1:(ln/2)+3)];
102 t_hs_3 = [t_hs_3((end-ln/2)-2:end) t_hs_3(1:(ln/2)+3)];
103 t_ls = t_ls((ln/2-ln_t/2)+1:ln/2+ln_t/2);
104 t_hs = t_hs((ln/2-ln_t/2)+1:ln/2+ln_t/2);
105 t_ls_2 = t_ls_2((ln/2-ln_t)+1:ln/2+ln_t);
106 t_hs_2 = t_hs_2((ln/2-ln_t)+1:ln/2+ln_t);
107 t_ls_3 = t_ls_3((ln/2-ln_t*2)+1:ln/2+ln_t*2);
108 t_hs_3 = t_hs_3((ln/2-ln_t*2)+1:ln/2+ln_t*2);
109 f_lp = (fftshift(f_lp)); %scale l=1 synthesis filters
110 f_hp = (fftshift(f_hp));
111 f_lp_2 = (fftshift(f_lp_2)); %scale l=2 synthesis filters
112 f_hp_2 = (fftshift(f_hp_2));
113 f_lp_3 = (fftshift(f_lp_3)); %scale l=2 synthesis filters
114 f_hp_3 = (fftshift(f_hp_3));
115 %=====================================================================
116 %---WAVELET RAMP FILTERS-----------------------------------------
117 %=====================================================================
118 w = -order:order-1;
119 for i=1:(length(angles))
```

```
120     uu(i,:) = costheta(i).*w;
121     vv(i,:) = sintheta(i).*w;
122 end
123 u = floor(uu)+order+1;
124 v = floor(vv)+order+1;
125 %L1 Filters
126 ml = f_lp(u);
127 mh = f_hp(u);
128 nl = f_lp(v);
129 nh = f_hp(v);
130 ml(:,1:order-1) = [];
131 mh(:,1:order-1) = [];
132 nl(:,1:order-1) = [];
133 nh(:,1:order-1) = [];
134 nl = conj(nl);
135 nh = conj(nh);
136 %L2 Filters
137 ml_2 = f_lp_2(u);
138 mh_2 = f_hp_2(u);
139 nl_2 = f_lp_2(v);
140 nh_2 = f_hp_2(v);
141 ml_2(:,1:order-1) = [];
142 mh_2(:,1:order-1) = [];
143 nl_2(:,1:order-1) = [];
144 nh_2(:,1:order-1) = [];
145 nl_2 = conj(nl_2);
146 nh_2 = conj(nh_2);
147 %L3 Filters
148 ml_3 = f_lp_3(u);
149 mh_3 = f_hp_3(u);
150 nl_3 = f_lp_3(v);
151 nh_3 = f_hp_3(v);
152 ml_3(:,1:order-1) = [];
153 mh_3(:,1:order-1) = [];
154 nl_3(:,1:order-1) = [];
155 nh_3(:,1:order-1) = [];
156 nl_3 = conj(nl_3);
157 nh_3 = conj(nh_3);
158 for i = 1:length(angles) %wavelet modified ramp filters
159     hor_filt(i,:) = ml(i,:).*nh(i,:).*filt;  % for every scale
160     vert_filt(i,:) = mh(i,:).*nl(i,:).*filt;
161     diag_filt(i,:) = mh(i,:).*nh(i,:).*filt;
```

```
162     hor_filt_2(i,:) = ml_2(i,:).*nh_2(i,:).*filt;
163     vert_filt_2(i,:) = mh_2(i,:).*nl_2(i,:).*filt;
164     diag_filt_2(i,:) = mh_2(i,:).*nh_2(i,:).*filt;
165     app_filt_3(i,:) = ml_3(i,:).*nl_3(i,:).*filt;
166     hor_filt_3(i,:) = ml_3(i,:).*nh_3(i,:).*filt;
167     vert_filt_3(i,:) = mh_3(i,:).*nl_3(i,:).*filt;
168     diag_filt_3(i,:) = mh_3(i,:).*nh_3(i,:).*filt;
169 end
170 %=====================================================================
171 %---COMPUTATION OF SCALE 1 BLOCK IMAGES-----------------------
172 %=====================================================================
173 N = size(p,1);
174 center = floor((N+1)/2);
175 resolution = 2;
176 xleft = -center + 1;
177 xx = (1:resolution:N) - 1 + xleft;
178 xx = repmat(xx, N/resolution, 1);
179 ytop = center - 1;
180 yy = (N:-resolution:1).' - N + ytop;
181 yy = repmat(yy, 1, N/resolution);
182 i=1;
183 m = 1;
184 n = 1;  %generate the pixel grids for the
185 while i<(N/l)+1   %appropriate size and scale
186  for j=0:(size(yy,1)/sqrt(N/l)):size(yy,1)-(size(yy,1)/sqrt(N/l))
187   for ...
        k=0:(size(xx,1)/sqrt(N/l)):size(xx,1)-(size(xx,1)/sqrt(N/l))
188    x{i} = ...
          xx(j+1:(size(xx,1)/sqrt(N/l)*m),k+1:(size(xx,2)/sqrt(N/l)*n));
189    y{i} = ...
          yy(j+1:(size(yy,1)/sqrt(N/l)*m),k+1:(size(yy,2)/sqrt(N/l)*n));
190    i = i+1;
191    n = n+1;
192   end
193   m = m+1;
194   n = 1;
195  end
196  m = 1;
197 end
198 ctrIdx = ceil(len/2);
199 %memory allocation
200 for i=1:length(sngrm_roi)
```

```
201     sngrm_roi{i}(:,order*2) = 0;
202     ft_sngrm_roi{i} = zeros(size(sngrm_roi{i}));
203     ft_sngrm_roi_half{i} = ...
            zeros(size(ft_sngrm_roi{i}(:,1:order+1)));
204 end
205 hor_filtered = zeros(size(ft_sngrm_roi_half{1}));
206 vert_filtered = zeros(size(ft_sngrm_roi_half{1}));
207 diag_filtered = zeros(size(ft_sngrm_roi_half{1}));
208 hor_filt_all = zeros(size(ft_sngrm_roi{1}));
209 vert_filt_all = zeros(size(ft_sngrm_roi{1}));
210 diag_filt_all = zeros(size(ft_sngrm_roi{1}));
211 w_hor_half = zeros(size(sngrm));
212 w_vert_half = zeros(size(sngrm));
213 w_diag_half = zeros(size(sngrm));
214 thor = zeros(1,size(sngrm_roi,2));
215 tvert = zeros(1,size(sngrm_roi,2));
216 tdiag = zeros(1,size(sngrm_roi,2));
217 t_proj = zeros(1,size(sngrm_roi{n},2));
218 t = zeros(size(x{1}));
219 a = zeros(size(x{1}));
220 for i=1:length(sngrm_roi)
221     imgd{i} = zeros((N/sqrt(N/l))/resolution,class(p));
222     imgv{i} = zeros((N/sqrt(N/l))/resolution,class(p));
223     imgh{i} = zeros((N/sqrt(N/l))/resolution,class(p));
224     imgd_nrm{i} = zeros((N/sqrt(N/l))/resolution,class(p));
225     imgv_nrm{i} = zeros((N/sqrt(N/l))/resolution,class(p));
226     imgh_nrm{i} = zeros((N/sqrt(N/l))/resolution,class(p));
227 end
228 for n = 1:length(sngrm_roi)
229     w_diag = zeros(size(hor_filt_all)); %clear arrays at ...
            every iteration
230     w_vert = zeros(size(hor_filt_all));
231     w_hor = zeros(size(hor_filt_all));
232 %=================================================================
233 %---FT OF PROJECTIONS---------------------------------------------
234 %=================================================================
235     for i =1: length(angles)
236         ft_sngrm_roi{n}(i,:) = (fft(sngrm_roi{n}(i,:)));
237     end
238     ft_sngrm_roi_half{n} = (ft_sngrm_roi{n}(:,1:order+1));
239 %=================================================================
240 %---FILTERING-----------------------------------------------------
```

```
%=========================================================================
    for i = 1:length(angles)
        hor_filtered(i,:) = ...
            ft_sngrm_roi_half{n}(i,:).*(hor_filt(i,:));
        vert_filtered(i,:) = ...
            ft_sngrm_roi_half{n}(i,:).*(vert_filt(i,:));
        diag_filtered(i,:) = ...
            ft_sngrm_roi_half{n}(i,:).*(diag_filt(i,:));
    end
    hor_filt_all = ([hor_filtered ...
        conj(hor_filtered(:,end-1:-1:2))]);
    vert_filt_all = ([vert_filtered ...
        conj(vert_filtered(:,end-1:-1:2))]);
    diag_filt_all = ([diag_filtered ...
        conj(diag_filtered(:,end-1:-1:2))]);
    for i = 1:length(angles)
        w_diag(i,:) = real(ifft(diag_filt_all(i,:)));
        w_vert(i,:) = real(ifft(vert_filt_all(i,:)));
        w_hor(i,:) = real(ifft(hor_filt_all(i,:)));
    end
    w_diag(:,(len)+1:end) = [];
    w_vert(:,(len)+1:end) = [];
    w_hor(:,(len)+1:end) = [];
%=========================================================================
%---BACKPROJECTION-------------------------------------------------
%=========================================================================
%diagonal details------
    for i=7:84  %involved angles
        t_proj = w_diag(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
        imgd{n} = imgd{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
        imgd_nrm{n} = imgd{n}*(pi/(length(theta)));
    end
    for i=98:178  %involved angles
        t_proj = w_diag(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
        imgd{n} = imgd{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
        imgd_nrm{n} = imgd{n}*(pi/(length(theta)));
```

```
275     end
276 %vertical details------
277     for i=1:85 %involved angles
278         t_proj = w_vert(i,:);
279         t = x{n}.*costheta(i) + y{n}.*sintheta(i);
280         a = floor(t);
281         imgv{n} = imgv{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
                (a+1-t).*t_proj(a+ctrIdx);
282         imgv_nrm{n} = imgv{n}*(pi/(length(theta)));
283     end
284     for i=97:180 %involved angles
285         t_proj = w_vert(i,:);
286         t = x{n}.*costheta(i) + y{n}.*sintheta(i);
287         a = floor(t);
288         imgv{n} = imgv{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
                (a+1-t).*t_proj(a+ctrIdx);
289         imgv_nrm{n} = imgv{n}*(pi/(length(theta)));
290     end
291 %horizontal details------
292     for i=6:176 %involved angles
293         t_proj = w_hor(i,:);
294         t = x{n}.*costheta(i) + y{n}.*sintheta(i);
295         a = floor(t);
296         imgh{n} = imgh{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
                (a+1-t).*t_proj(a+ctrIdx);
297         imgh_nrm{n} = imgh{n}*(pi/(length(theta)));
298     end
299 end
300 %=================================================================
301 %---COMPUTATION OF SCALE 2 BLOCK IMAGES-----------------------
302 %=================================================================
303 resolution = 4;
304 xleft = -center + 1;
305 xx = (1:resolution:N) - 1 + xleft;
306 xx = repmat(xx, N/resolution, 1);
307 ytop = center - 1;
308 yy = (N:-resolution:1).' - N + ytop;
309 yy = repmat(yy, 1, N/resolution);
310 i=1;
311 m = 1;
312 n = 1;
313 while i<(N/l_2)+1
```

```
314   for ...
          j=0:(size(yy,1)/sqrt(N/l_2)):size(yy,1)-(size(yy,1)/sqrt(N/l_2))
315    for ...
           k=0:(size(xx,1)/sqrt(N/l_2)):size(xx,1)-(size(xx,1)/sqrt(N/l_2))
316     x{i} = ...
            xx(j+1:(size(xx,1)/sqrt(N/l_2)*m),k+1:(size(xx,2)/sqrt(N/l_2)*n));
317     y{i} = ...
            yy(j+1:(size(yy,1)/sqrt(N/l_2)*m),k+1:(size(yy,2)/sqrt(N/l_2)*n));
318     i = i+1;
319     n = n+1;
320    end
321    m = m+1;
322    n = 1;
323   end
324   m = 1;
325 end
326 ctrIdx = ceil(len/2);
327 %zero-padding
328 for i=1:length(sngrm_roi_2)
329     sngrm_roi_2{i}(:,order*2) = 0;
330     ft_sngrm_roi_2{i} = zeros(size(sngrm_roi_2{i}));
331     ft_sngrm_roi_half{i} = ...
            zeros(size(ft_sngrm_roi_2{i}(:,1:order+1)));
332 end
333 hor_filtered = zeros(size(ft_sngrm_roi_half{1}));
334 vert_filtered = zeros(size(ft_sngrm_roi_half{1}));
335 diag_filtered = zeros(size(ft_sngrm_roi_half{1}));
336 hor_filt_all = zeros(size(ft_sngrm_roi_2{1}));
337 vert_filt_all = zeros(size(ft_sngrm_roi_2{1}));
338 diag_filt_all = zeros(size(ft_sngrm_roi_2{1}));
339 w_hor_half = zeros(size(sngrm));
340 w_vert_half = zeros(size(sngrm));
341 w_diag_half = zeros(size(sngrm));
342 thor = zeros(1,size(sngrm_roi_2,2));
343 tvert = zeros(1,size(sngrm_roi_2,2));
344 tdiag = zeros(1,size(sngrm_roi_2,2));
345 t_proj = zeros(1,size(sngrm_roi_2{n},2));
346 t = zeros(size(x{1}));
347 a = zeros(size(x{1}));
348 for i=1:length(sngrm_roi_2)
349     imgd_2{i} = zeros((N/sqrt(N/l_2))/resolution,class(p));
350     imgv_2{i} = zeros((N/sqrt(N/l_2))/resolution,class(p));
```

```
351     imgh_2{i} = zeros((N/sqrt(N/l_2))/resolution,class(p));
352     imgd_2_nrm{i} = zeros((N/sqrt(N/l_2))/resolution,class(p));
353     imgv_2_nrm{i} = zeros((N/sqrt(N/l_2))/resolution,class(p));
354     imgh_2_nrm{i} = zeros((N/sqrt(N/l_2))/resolution,class(p));
355 end
356 for n = 1:length(sngrm_roi_2)
357     w_diag = zeros(size(hor_filt_all));
358     w_vert = zeros(size(hor_filt_all));
359     w_hor = zeros(size(hor_filt_all));
360 %=========================================================================
361 %---FT OF PROJECTIONS-----------------------------------------
362 %=========================================================================
363     for i =1: length(angles)
364         ft_sngrm_roi_2{n}(i,:) = fft(sngrm_roi_2{n}(i,:));
365     end
366     ft_sngrm_roi_half{n} = ft_sngrm_roi_2{n}(:,1:order+1);
367 %=========================================================================
368 %---FILTERING-------------------------------------------------
369 %=========================================================================
370     for i = 1:length(angles)
371         hor_filtered(i,:) = ...
              ft_sngrm_roi_half{n}(i,:).*hor_filt_2(i,:);
372         vert_filtered(i,:) = ...
              ft_sngrm_roi_half{n}(i,:).*vert_filt_2(i,:);
373         diag_filtered(i,:) = ...
              ft_sngrm_roi_half{n}(i,:).*diag_filt_2(i,:);
374     end
375     hor_filt_all = ([hor_filtered ...
         conj(hor_filtered(:,end-1:-1:2))]);
376     vert_filt_all = ([vert_filtered ...
         conj(vert_filtered(:,end-1:-1:2))]);
377     diag_filt_all = ([diag_filtered ...
         conj(diag_filtered(:,end-1:-1:2))]);
378     for i = 1:length(angles)
379         w_diag(i,:) = real(ifft(diag_filt_all(i,:)));
380         w_vert(i,:) = real(ifft(vert_filt_all(i,:)));
381         w_hor(i,:) = real(ifft(hor_filt_all(i,:)));
382     end
383     w_diag(:,(len)+1:end) = [];
384     w_vert(:,(len)+1:end) = [];
385     w_hor(:,(len)+1:end) = [];
386
```

```
%=========================================================================
%---BACKPROJECTION-------------------------------------------------------
%=========================================================================
%diagonal details------
    for i=4:87
        t_proj = w_diag(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
        imgd_2{n} = imgd_2{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
        imgd_2_nrm{n} = imgd_2{n}*(pi/(length(theta)));
    end
    for i=95:177
        t_proj = w_diag(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
        imgd_2{n} = imgd_2{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
        imgd_2_nrm{n} = imgd_2{n}*(pi/(length(theta)));
    end
%vertical details------
    for i=1:86
        t_proj = w_vert(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
        imgv_2{n} = imgv_2{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
        imgv_2_nrm{n} = imgv_2{n}*(pi/(length(theta)));
    end
    for i=96:180
        t_proj = w_vert(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
        imgv_2{n} = imgv_2{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
        imgv_2_nrm{n} = imgv_2{n}*(pi/(length(theta)));
    end
%diagonal details------
    for i=5:175
        t_proj = w_hor(i,:);
        t = x{n}.*costheta(i) + y{n}.*sintheta(i);
        a = floor(t);
```

```
425         imgh_2{n} = imgh_2{n} + (t-a).*t_proj(a+1+ctrIdx) + ...
                (a+1-t).*t_proj(a+ctrIdx);
426         imgh_2_nrm{n} = imgh_2{n}*(pi/(length(theta)));
427     end
428 end
429 %=================================================================
430 %---COMPUTATION OF SCALE 3 BLOCK IMAGES-----------------------
431 %=================================================================
432 resolution = 8;
433 xleft = -center + 1;
434 x = (1:resolution:N) - 1 + xleft;
435 x = repmat(x, N/resolution, 1);
436 ytop = center - 1;
437 y = (N:-resolution:1).' - N + ytop;
438 y = repmat(y, 1, N/resolution);
439 ctrIdx = ceil(len/2);
440 %zero-padding
441 sngrm_roi_3(:,order*2) = 0;
442 ft_sngrm_roi_3 = zeros(size(sngrm_roi_3));
443 ft_sngrm_roi_half = zeros(size(ft_sngrm_roi_3(:,1:order+1)));
444 hor_filtered = zeros(size(ft_sngrm_roi_half));
445 vert_filtered = zeros(size(ft_sngrm_roi_half));
446 diag_filtered = zeros(size(ft_sngrm_roi_half));
447 app_filtered = zeros(size(ft_sngrm_roi_half));
448 hor_filt_all = zeros(size(ft_sngrm_roi_3));
449 vert_filt_all = zeros(size(ft_sngrm_roi_3));
450 diag_filt_all = zeros(size(ft_sngrm_roi_3));
451 app_filt_all = zeros(size(ft_sngrm_roi_3));
452 w_hor_half = zeros(size(sngrm));
453 w_vert_half = zeros(size(sngrm));
454 w_diag_half = zeros(size(sngrm));
455 w_app_half = zeros(size(sngrm));
456 thor = zeros(1,size(sngrm_roi_3,2));
457 tvert = zeros(1,size(sngrm_roi_3,2));
458 tdiag = zeros(1,size(sngrm_roi_3,2));
459 tapp = zeros(1,size(sngrm_roi_3,2));
460 t_proj = zeros(1,size(sngrm_roi_3,2));
461 t = zeros(size(x));
462 a = zeros(size(x));
463 imgd_3 = zeros(N/resolution,class(p));
464 imgv_3 = zeros(N/resolution,class(p));
465 imgh_3 = zeros(N/resolution,class(p));
```

```
466 imga_3 = zeros(N/resolution,class(p));
467 imgd_3_nrm = zeros(N/resolution,class(p));
468 imgv_3_nrm = zeros(N/resolution,class(p));
469 imgh_3_nrm = zeros(N/resolution,class(p));
470 imga_3_nrm = zeros(N/resolution,class(p));
471 w_diag = zeros(size(hor_filt_all));
472 w_vert = zeros(size(hor_filt_all));
473 w_hor = zeros(size(hor_filt_all));
474 w_app = zeros(size(hor_filt_all));
475 %=================================================================
476 %---FT OF PROJECTIONS-----------------------------------------
477 %=================================================================
478 for i =1: length(angles)
479     ft_sngrm_roi_3(i,:) = (fft(sngrm_roi_3(i,:)));
480 end
481 ft_sngrm_roi_half = ft_sngrm_roi_3(:,1:order+1);
482 %=================================================================
483 %---FILTERING-------------------------------------------------
484 %=================================================================
485 for i = 1:length(angles)
486     hor_filtered(i,:) = ...
          ft_sngrm_roi_half(i,:).*(hor_filt_3(i,:));
487     vert_filtered(i,:) = ...
          ft_sngrm_roi_half(i,:).*(vert_filt_3(i,:));
488     diag_filtered(i,:) = ...
          ft_sngrm_roi_half(i,:).*(diag_filt_3(i,:));
489     app_filtered(i,:) = ...
          ft_sngrm_roi_half(i,:).*(app_filt_3(i,:));
490 end
491 hor_filt_all = ([hor_filtered conj(hor_filtered(:,end-1:-1:2))]);
492 vert_filt_all = ([vert_filtered ...
      conj(vert_filtered(:,end-1:-1:2))]);
493 diag_filt_all = ([diag_filtered ...
      conj(diag_filtered(:,end-1:-1:2))]);
494 app_filt_all = ([app_filtered conj(app_filtered(:,end-1:-1:2))]);
495 for i = 1:length(angles)
496     w_diag(i,:) = real(ifft(diag_filt_all(i,:)));
497     w_vert(i,:) = real(ifft(vert_filt_all(i,:)));
498     w_hor(i,:) = real(ifft(hor_filt_all(i,:)));
499     w_app(i,:) = real(ifft(app_filt_all(i,:)));
500 end
501 w_diag(:,(len)+1:end) = [];
```

```
w_vert(:,(len)+1:end) = [];
w_hor(:,(len)+1:end) = [];
w_app(:,(len)+1:end) = [];
%=================================================================
%---BACKPROJECTION-----------------------------------------------
%=================================================================
%diagonal details---------
for i=3:88
    t_proj = w_diag(i,:);
    t = x.*costheta(i) + y.*sintheta(i);
    a = floor(t);
    imgd_3 = imgd_3 + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imgd_3_nrm = imgd_3*(pi/(length(theta)));
end
for i=94:179
    t_proj = w_diag(i,:);
    t = x.*costheta(i) + y.*sintheta(i);
    a = floor(t);
    imgd_3 = imgd_3 + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imgd_3_nrm = imgd_3*(pi/(length(theta)));
end
%vertical details---------
for i=1:87
    t_proj = w_vert(i,:);
    t = x.*costheta(i) + y.*sintheta(i);
    a = floor(t);
    imgv_3 = imgv_3 + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imgv_3_nrm = imgv_3*(pi/(length(theta)));
end
for i=95:180
    t_proj = w_vert(i,:);
    t = x.*costheta(i) + y.*sintheta(i);
    a = floor(t);
    imgv_3 = imgv_3 + (t-a).*t_proj(a+1+ctrIdx) + ...
        (a+1-t).*t_proj(a+ctrIdx);
    imgv_3_nrm = imgv_3*(pi/(length(theta)));
end
%horizontal details---------
for i=4:187
```

```
540     t_proj = w_hor(i,:);
541     t = x.*costheta(i) + y.*sintheta(i);
542     a = floor(t);
543     imgh_3 = imgh_3 + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
544     imgh_3_nrm = imgh_3*(pi/(length(theta)));
545 end
546 %approximations---------
547 for i=1:length(angles)
548     t_proj = w_app(i,:);
549     t = x.*costheta(i) + y.*sintheta(i);
550     a = floor(t);
551     imga_3 = imga_3 + (t-a).*t_proj(a+1+ctrIdx) + ...
            (a+1-t).*t_proj(a+ctrIdx);
552     imga_3_nrm = imga_3*(pi/(length(theta)));
553 end
554 %================================================================
555 %---COEFFICIENT IMAGES BLOCK TILING-------------------------
556 %================================================================
557 imagend = zeros(128); %Scale 1
558 imagenv = zeros(128);
559 imagenh = zeros(128);
560 cc = 1;
561 n = 1;
562 for j = 1 : (size(imagend,1)/4) : size(imagend,1)
563  c = 1;
564  for i = 1 : (size(imagend,1)/4) : size(imagend,1)
565   imagend((j:cc*size(imagend,1)/4),(i:c*size(imagend,1)/4)) = ...
          imgd_nrm{n};
566   imagenv((j:cc*size(imagenv,1)/4),(i:c*size(imagenv,1)/4)) = ...
          imgv_nrm{n};
567   imagenh((j:cc*size(imagenh,1)/4),(i:c*size(imagenh,1)/4)) = ...
          imgh_nrm{n};
568   c = c+1;
569   n = n+1;
570  end
571  cc = cc + 1;
572 end
573 imagend_2 = zeros(64); %Scale 2
574 imagenv_2 = zeros(64);
575 imagenh_2 = zeros(64);
576 cc = 1;
```

```
n = 1;
for j = 1 : (size(imagend_2,1)/2) : size(imagend_2,1)
 c = 1;
 for i = 1 : (size(imagend_2,1)/2) : size(imagend_2,1)
  imagend_2((j:cc*size(imagend_2,1)/2),(i:c*size(imagend_2,1)/2)) ...
     = imgd_2_nrm{n};
  imagenv_2((j:cc*size(imagenv_2,1)/2),(i:c*size(imagenv_2,1)/2)) ...
     = imgv_2_nrm{n};
  imagenh_2((j:cc*size(imagenh_2,1)/2),(i:c*size(imagenh_2,1)/2)) ...
     = imgh_2_nrm{n};
  c = c+1;
  n = n+1;
 end
 cc = cc + 1;
end
imagend_3 = imgd_3_nrm; %Scale 3
imagenv_3 = imgv_3_nrm;
imagenh_3 = imgh_3_nrm;
imagena_3 = imga_3_nrm;
%=================================================================
%---WAVELET RECONSTRUCTION L3-------------------------------
%=================================================================
lp = t_ls_3;
hp = t_hs_3;
%Upsampling--------------------------
rdiag_3 = upsample((imagend_3), 8, 3);
rvert_3 = upsample((imagenv_3), 8, 3);
rhor_3 = upsample((imagenh_3), 8, 3);
rapp_3 = upsample((imagena_3), 8, 3);
rdiag_3 = rdiag_3.';
rvert_3 = rvert_3.';
rhor_3 = rhor_3.';
rapp_3 = rapp_3.';
rdiag_3 = upsample(rdiag_3, 8, 3);
rvert_3 = upsample(rvert_3, 8, 3);
rhor_3 = upsample(rhor_3, 8, 3);
rapp_3 = upsample(rapp_3, 8, 3);
rdiag_3 = rdiag_3.';
rvert_3 = rvert_3.';
rhor_3 = rhor_3.';
rapp_3 = rapp_3.';
rdiag_3 = conv2(rdiag_3, hp.', 'same');
```

```
616 rvert_3 = conv2(rvert_3, lp.', 'same');
617 rhor_3 = conv2(rhor_3, hp.', 'same');
618 rapp_3 = conv2(rapp_3, lp.', 'same');
619 rdiag_3 = conv2(rdiag_3, hp, 'same');
620 rvert_3 = conv2(rvert_3, hp, 'same');
621 rhor_3 = conv2(rhor_3, lp, 'same');
622 rapp_3 = conv2(rapp_3, lp, 'same');
623 image_3 = (rapp_3 + rdiag_3 + rvert_3 + rhor_3);
624 %============================================================
625 %---WAVELET RECONSTRUCTION L2-----------------------------
626 %============================================================
627 lp = t_ls_2;
628 hp = t_hs_2;
629 %Upsampling--------------------------
630 rdiag_2 = upsample((imagend_2), 4, 1);
631 rvert_2 = upsample((imagenv_2), 4, 1);
632 rhor_2 = upsample((imagenh_2), 4, 1);
633 rdiag_2 = rdiag_2.';
634 rvert_2 = rvert_2.';
635 rhor_2 = rhor_2.';
636 rdiag_2 = upsample(rdiag_2, 4, 1);
637 rvert_2 = upsample(rvert_2, 4, 1);
638 rhor_2 = upsample(rhor_2, 4, 1);
639 rdiag_2 = rdiag_2.';
640 rvert_2 = rvert_2.';
641 rhor_2 = rhor_2.';
642 rdiag_2 = conv2(rdiag_2, hp.', 'same');
643 rvert_2 = conv2(rvert_2, lp.', 'same');
644 rhor_2 = conv2(rhor_2, hp.', 'same');
645 rdiag_2 = conv2(rdiag_2, hp, 'same');
646 rvert_2 = conv2(rvert_2, hp, 'same');
647 rhor_2 = conv2(rhor_2, lp, 'same');
648 image_2 = (rdiag_2 + rvert_2 + rhor_2);
649 %============================================================
650 %---WAVELET RECONSTRUCTION L1-----------------------------
651 %============================================================
652 lp = t_ls;
653 hp = t_hs;
654 %Upsampling--------------------------
655 rdiag = upsample((imagend), 2);
656 rvert = upsample((imagenv), 2);
657 rhor = upsample((imagenh), 2);
```

```
658 rdiag = rdiag.';
659 rvert = rvert.';
660 rhor = rhor.';
661 rdiag = upsample(rdiag, 2);
662 rvert = upsample(rvert, 2);
663 rhor = upsample(rhor, 2);
664 rdiag = rdiag.';
665 rvert = rvert.';
666 rhor = rhor.';
667 rdiag = conv2(rdiag, hp.', 'same');
668 rvert = conv2(rvert, lp.', 'same');
669 rhor = conv2(rhor, hp.', 'same');
670 rdiag = conv2(rdiag, hp, 'same');
671 rvert = conv2(rvert, hp, 'same');
672 rhor = conv2(rhor, lp, 'same');
673 image = (rdiag + rvert + rhor);
674 imagen = (image + image_2 + image_3); %Output Image
675
676 [global_sim local_sim] = ssim(imagen,p);
677 figure, imshow(local_sim,[])
678
679 PSNR = psnr(imagen, p)
```

A.5.1 Block Sinogram Templates Function for Projection Data Truncation

```
1  function [ ] = sngrm_templates(N)
2  %=================================================================
3  %BLOCK IMAGES TEMPLATES FOR TRUNCATION IN RADON DOMAIN-----------
4  %=================================================================
5  p = phantom(N);
6  angles = 0:179;
7  %---load block-support phantom images--------------
8  for i = 1:16 %Scale 1
9      filename = sprintf('templates_32x32_16_20pc/%d.png',i); %path
10     template_1{i} = imread(filename);
11     template_1{i} = logical(template_1{i}(:,:,1)); %RGB to double
12     sngrm_template_1{i} = radon(template_1{i}, angles);%blck ...
           snrgm
13 end
```

```
for i = 1:4 %Scale 2
    filename = sprintf('templates_8x8_4_30pc/%d.png',i);
    template_2{i} = imread(filename);
    template_2{i} = logical(template_2{i}(:,:,1));
    sngrm_template_2{i} = radon(template_2{i}, angles);
end
%---generate logic-valued sinogram tamplates
for i = 1:size(sngrm_template_1,2)
    grid_template_1{i} = logical(sngrm_template_1{i});
    filenames = sprintf('sinograms64/template_1_%d.png',i);
    imwrite(grid_template_1{i}, filenames);
end
for i = 1:size(sngrm_template_2,2)
    grid_template_2{i} = logical(sngrm_template_2{i});
    filenames = sprintf('sinograms64/template_2_%d.png',i);
    imwrite(grid_template_2{i}, filenames);
end
```

Appendix B

Sensors Journal Draft Manuscript

Tiled-Block Image Reconstruction by Wavelet-Based, Parallel Filtered Back-Projection

Jorge Guevara Escobedo and Krikor B. Ozanyan, *Senior Member, IEEE*

***Abstract*— We demonstrate an algorithm, relevant to tomography sensor systems, to obtain images from the parallel reconstruction of essentially localized elements at different scales. This is achieved by combining methodology to reconstruct images from limited and/or truncated data, with the time-frequency capabilities of the Wavelet Transform. Multiscale, as well as time-frequency, localization properties of the separable two-dimensional wavelet transform are exploited as an approach for faster reconstruction. The speed up is realized not only by reducing the computation load on a single processor, but also by achieving the parallel reconstruction of several tiled blocks. With tiled-block image reconstruction by wavelet-based, parallel filtered back-projection we measure more than 36 times gain in speed, compared to standard filtered backprojection.**

***Index Terms*— image reconstruction, computed tomography, data processing algorithms, parallel processing, discrete wavelet transform.**

I. Introduction

ACROSS many applications, it is common to identify the need of information about an object, without altering its physical structure. Fortunately, there are numerous methods allowing radiation, either emitted or transmitted, to be employed to obtain cross-sectional images characterizing the inner structure of an object. The mathematical foundation behind such an approach was developed by Johann Radon in 1917 [1] and several decades later, in 1972, experimentally implemented by Hounsfield [2] resulting in the demonstration of the first Computed Tomography (CT) scanner.

The main motivation for this work was the existing body of knowledge and achievements on the reconstruction of reduced-area full-resolution images, originally encouraged by the radiation dose exposure reduction in medical imaging, and where the Wavelet Transform (WT), along its different representations, proved to be an effective tool given its time-frequency localization capabilities [3]-[4]. Such research has been commonly named as Wavelet-based local reconstruction, and has been reported to be useful in other application areas such as Nano and Micro Tomography [5]-[6]; Terahertz Tomography [7], and Dental Radiology [8].

Of special interest is the 2D separable representation of the WT, employed in [4] and [9]. In addition to achieving local reconstruction, it allows projection data to be processed in a multiresolution scheme. Such a feature proves to be of great benefit, when realizing its similarities with the parallel algorithm proposed in [10]. In that algorithm, a frequential decomposition of projection data is performed with the aim to back-project every component separately. The subsequent merger into a final image notably speeds up the image reconstruction process.

In this work we propose an algorithm combining the wavelet-based reconstruction of reduced image areas [9],[4]; and the parallel reconstruction from frequential sub-band decomposition of projection data [10]. The outline of the paper is as follows: Section II covers the background theory involved in the algorithm development; tomography image reconstruction, the wavelet transform and the multiresolution representation. Section III details the implementation of the algorithm. The results are shown in section IV and the functionality of the approach is discussed in section V.

II. Background

A. Image Reconstruction from Projection Data.

In Tomography, as well as in other imaging techniques, the Radon Transform (RT) is the mathematical tool employed to map an unknown density distribution (object) (x,y) onto attenuation line integrals passing across it. In 2D, the analytical expression yielding the set of parallel line integrals is given by:

$$g(\theta,r) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - r)dxdy \quad (1)$$

where θ is the angle between the line normal and the x-axis, and r the distance from the rotation origin. Data collected (e.g. by a detector array) at the same angle θ are grouped in a projection and the set of projections taken at all angles constitutes the sampled RT of the object. The 2D graphical representation of RT (the "sinogram") is given by a stack of all projections ordered by angle, each projection contributing with a row of pixel intensities determined by the values of the constituent line integrals.

An important property, utilized to solve the inverse problem of recovering $f(x,y)$ from projection data, is given by the Fourier Slice theorem, stating that the values calculated by the 1D Fourier transform $G(\theta,\omega)$ of a projection at an angle θ:

$$\mathcal{F}_{1D}[g(\theta,r)] = G(\theta,\omega), \quad (2)$$

populate a "diagonal slice" at the same angle θ within the 2D Fourier image $F(u,v)$ of the object. Therefore, by taking the Fourier transform of the measured projection data at a sufficient amount of angles to assemble a well sampled

Manuscript received October 15, 2015.

Jorge Guevara Escobedo (jorge.guevaraescobedo@manchester.ac.uk) and Krikor B. Ozanyan (k.ozanyan@manchester.ac.uk) are with the School of Electrical and Electronic Engineering, The University of Manchester, M13 9PL, Manchester, UK. JGE wishes to acknowledge the financial support of Consejo Nacional de Ciencia y Tecnología (CONACyT), Mexico, for a doctoral studentship.

representation of the object's Fourier space, the solution to the inverse Radon problem is reduced to taking the 2D inverse transform

$$f(x,y) = \mathcal{F}_{2D}^{-1}[F(u,v)] \quad . \tag{3}$$

In a practical implementation, it is not realistic to obtain projection data that fully covers the Fourier space. Therefore, the Filtered Backprojection (FBP) method is used to account for adjustment of the integration limits in the Fourier inversion, along with a change from cartesian to polar coordinates. The FBP is mathematically defined by

$$f(x,y) = \int_0^{\pi} \int_{-\infty}^{\infty} G(\theta,\omega)|\omega| e^{j2\omega r}\, d\theta d\omega \ , \tag{4}$$

where $|\omega|$ is the Jacobian of the coordinate system transformation and acts as a ramp filter suppressing the low frequencies close to the origin. The above formula can be split into two steps: the filtering of projection data

$$Q(\theta,r) = \int_{-\infty}^{\infty} G(\theta,\omega)|\omega| e^{j2\omega r}\, d\omega \tag{5}$$

and the backprojection

$$f(x,y) = \int_0^{\pi} Q_{\theta}(x\cos\theta + y\sin\theta) d\theta \ , \tag{6}$$

where $x\cos\theta + y\sin\theta = r$.

A more detailed explanation and the strict derivation of the FBP expression are available in [12].

B. Global character (nonlocality) of the standard FBP

While the FBP attracts interest, due to its high computational efficiency [13], its functionality is limited by the availability of sufficient projection data, equispaced in θ and r [14]. In cases when access restrictions prevent the collection of complete data or measurements need to be limited only to a region of interest, (4) is not the most indicated reconstruction strategy. This is due to the non-locality of the standard FBP caused mainly by the ramp filter $|\omega|$, and particularly its discontinuity at the origin. The latter results in the spread of projection data support in the space domain.

This problem has been addressed under different names: i) Reconstruction from truncated projections [14], ii) Incomplete data problem [15], iii) Local Tomography [16] and iv) The interior Radon Transform [17] and v) Tomography from scarce measurements E Constantino, KB Ozanyan Sensors Journal, IEEE 9 (4), 399-410. All approaches serve the same purpose, to avoid the global dependency of the FBP and reconstruct an accurate image from compactly supported data. Different approaches to local or limited data reconstruction have been reviewed in [18]-[20], where the use of wavelet transforms emerges as one with the most promising methodology. This is due to the expectation that by employing a function having a sufficient amount of zero moments, the support will remain essentially unchanged. Wavelet filters are functions with compact support and can be constructed with a certain amount of zero moments [21]-[23].

C. The Wavelet Transform

The main idea behind the wavelet transform is to obtain information not only about the constituent frequencies of a signal, but also about the interval of time relevant to these frequencies.

As a cursory explanation, the continuous wavelet transform (CTW) is typically based on the complete temporal correlation between the input signal and dilated/expanded versions of a predefined finite length signal of the mother wavelet. A match between a certain frequency in a temporal segment from the input signal and the wavelet being shifted along that segment is characterized by a high correlation coefficient and the contribution of that frequency, along with the temporal parameters of the segment, is flagged [24]. However, the CWT involves redundant processing of large data sets, not suitable for practical implementation.

A different representation scheme was proposed in [25], whereby a discrete signal is processed with a range of high/low pass digital filters and down-samplers; the result is a decomposition of the input signal into scale components differing in size by a factor of two, to its nearest higher scale. Low and high pass filters (scaling functions (n)) and wavelet function $\psi(n)$ are convolved with the input signal $x(n)$ to produce a pair of wavelet coefficients: approximations (lower frequencies) and details (higher frequencies). After filtering, the output wavelet coefficients' bandwidth is reduced by half, therefore by means of the Shannon theorem, these coefficients can be represented with as half as many of the samples contained in the original signal, without any loss:

$$\begin{aligned} W_s(k) &= s(n) * x(n) \downarrow_{n=2k,k\geq 0} \\ W_{\psi}(k) &= \psi(n) * x(n) \downarrow_{n=2k,k\geq 0} \end{aligned} \ , \tag{7}$$

where * is the convolution symbol. To recover the original signal, the wavelet coefficients are up-sampled and reverse-filtered, resulting in two signals. The merger of these signals must be equivalent to the original signal, if the employed wavelet functions allow exact reconstruction. This is illustrated in Fig. 1.

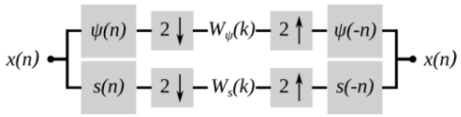


Fig. 1. Separable wavelet filter bank.

In a multiscale representation, the approximation coefficients (k) become the input to a new frequency decomposition and the output coefficients will again contain only half as many samples as before.

For a 2D input signal, three wavelet functions are obtained from separable products between both scaling and wavelet functions. They quantify high frequency variations along different directions: horizontal, vertical and diagonal.

$$\begin{aligned} \Psi^H(m,n) &= s(m)\psi(n) \quad \text{Horizontal} \\ \psi^V(m,n) &= \psi(m)s(n) \quad \text{Vertical} \\ \psi^D(m,n) &= \psi(m)\psi(n) \quad \text{Diagonal} \end{aligned} \tag{8}$$

And the scaling function is as follows:

$$s(m,n) = s(m)s(n) \tag{9}$$

Performing the 2D separable wavelet transform amounts then to the calculation of the 1D version along the rows and columns of the 2D input signal.

$$\begin{aligned} W_{\psi}^{H}(m,n) &= \psi(m) * [x(m,n) * s(n) \downarrow_{n=2k}] \downarrow_{m=2k} \\ W_{\psi}^{V}(m,n) &= s(m) * [x(m,n) * \psi(n) \downarrow_{n=2k}] \downarrow_{m=2k} \\ W_{\psi}^{D}(m,n) &= \psi(m) * [x(m,n) * \psi(n)) \downarrow_{n=2k}] \downarrow_{m=2k} \\ W_{s}(m,n) &= s(m) * [x(m,n) * s(n)) \downarrow_{n=2k}] \downarrow_{m=2k} \end{aligned} \tag{10}$$

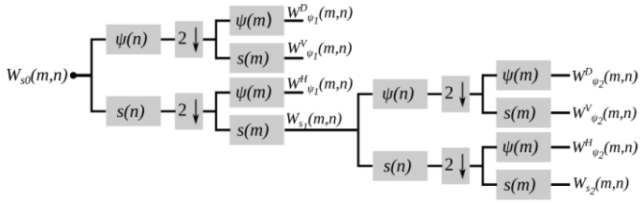


Fig. 2. Two-step separable two-dimensional wavelet transform.

where $W_{\psi^H}(m,n)$, $W_{\psi^V}(m,n)$, and $W_{\psi^D}(m,n)$ are detail coefficients along horizontal, vertical and diagonal directions. $W_S(m,n)$ is the approximations coefficient, $x(m,n)$ is the input signal. Analogous to the 1D case, a multi-scale representation is achieved by the approximations wavelet coefficient becoming the input for a new wavelet transform iteration, as shown in Fig. 2. Thus, 4 coefficients are obtained at each scale: three separate detail coefficients and one approximations coefficient, corresponding to the lower scale component.

D. Wavelet-based Localized Tomography Reconstruction

1) Wavelet-modified ramp filtering

The main motivation in involving the wavelet transform in the FBP process is to avoid the support spread of projection data that occurs after the ramp filtering, since the latter emphasizes the global dependency of the measurements' overall support. As suggested in [21], for the accurate reconstruction of a ROI, the filtering stage of the standard FBP can be extended with the addition of essentially compactly supported functions, presenting a certain amount of zero moments. The low/high pass filters that act as wavelet/scaling functions in the 2D WT, are functions of that type, and can be incorporated in (5) as follows.

$$W_{coeff}(x,y) = \int_0^{\pi}\int_{-\infty}^{\infty} G_{\theta}(\omega)[|\omega| W(\omega\cos\theta, \omega\sin\theta)]e^{j2\pi\omega s}d\omega d\theta \tag{11}$$

The above equation is the wavelet-based FBP, where $|\omega|(\omega cos\theta,\omega sin\theta)$ corresponds to the wavelet-modified ramp filter. W relates to the product between 1D high/low pass filters in Fourier domain and polar coordinates, being the last mentioned characteristic, necessary to match with the format of projection data. Separable products that correspond to each of the 2D WT image coefficients are shown below:

$$\begin{aligned} W(\omega\cos\theta, \omega\sin\theta) &= \\ S(\omega\cos\theta, \omega\sin\theta) &= S(\omega\cos\theta)\, S(\omega\sin\theta) \\ \Psi^{H}(\omega\cos\theta, \omega\sin\theta) &= S(\omega\cos\theta)\, \Psi(\omega\sin\theta) \\ \Psi^{V}(\omega\cos\theta, \omega\sin\theta) &= \Psi(\omega\cos\theta)\, S(\omega\sin\theta) \\ \Psi^{D}(\omega\cos\theta, \omega\sin\theta) &= \Psi(\omega\cos\theta)\, \Psi(\omega\sin\theta) \end{aligned} \tag{12}$$

It can be noticed from (12) that because of the cartesian to polar re-gridding of low/high filter coefficients, The wavelet-modified ramp filters become angle dependent. Such dependency suffers the drawback that filter coefficients must be calculated for every projection angle separately. However, it also presents with the great advantage that the detail frequency components at a number of angles can be discarded because of their null effect in the projection data filtering. Unfortunately, this is not the case for the approximations frequency component, which incorporates all lower frequencies in all projection angles.

2) The backprojection operator and subsampling

It can be seen from (12) that unlike the standard FBP the filtering in the wavelet-based FBP generalization shifts the approach, from the frequency weighting of projection data within the Fourier space, to the decomposition into frequency components, as previously explained in section II C.. This implies that backprojection has to be performed to create four different wavelet coefficient images, each of them half the size of the image yielded by the standard FBP. A remarkable consequence of such a characteristic is that the complexity involved in this operation is considerably reduced when implementing it parallel computation.

III. THE PARALLEL MULTISCALE WAVELET RECONSTRUCTION

A. Methodology

In the approach proposed in [10], projection data is mapped onto a Fourier space divided into *BxB* adjacent squares with the objective to parallelize the reconstruction process. These squares define certain references to process only projection data that concerns them. The result is a set of *BxB* frequency components which can be, independently and in parallel, brackprojected onto less dense grids that are finally merged to form a single output image. By observing on Fig.3 how the 2D WT tiles the Fourier space, it can be concluded that very

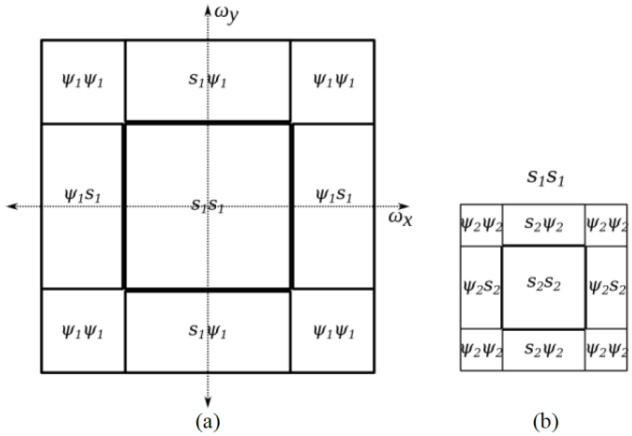


Fig. 3. Wavelet decomposition of the Fourier space. (a) One-step scale decomposition. (b) Two-step scale decomposition.

similar properties are already implicit in the wavelet-based FBP. The main difference lies in the manner with which the combination of wavelet and scaling functions tiles the Fourier space into a fixed number of regions defining the frequencies that constitute each of the wavelet coefficients: three details and one approximations, as illustrated in Fig. 3(a).

If the approximations coefficient is used as the input of a new 2D WT iteration, as shown in Fig. 3(b), the Fourier space can be tiled into more lower-scale regions, substituting the approximations coefficient by a set of new lower-scale wavelet coefficients. Although it is expected that a multiscale decomposition contributes to the speed-up of the reconstruction process, the 2D WT in its standard form, suffers from the requirement for the iterative computation of approximations at each scale.

This limitation can be bypassed through the use of Noble identities [26], which allow the construction of an equivalent parallel multiscale implementation of the 2D WT in Fourier domain. The result is the substitution of the pyramidal configuration shown in Fig. 2, formed by low/high pass filters and downsamplers, into a scale-dependent set of filters and downsamplers. According to [9], for a non-pyramidal implementation, the low pass filter coefficients in the Fourier domain are as follows:

$$s_l(n) \leftrightarrow S_l(z) = \begin{cases} \prod_{q=0}^{l-1} H(2^q z) & l = 1,2,3\cdots \\ 1 & l = 0 \end{cases}, \quad (13)$$

and for high pass filters:

$$\psi_l(n) \leftrightarrow \Psi_l(z) = \begin{cases} \Psi(2^{l-1}z) S_{l-1}(z) & l = 1,2,3\cdots \\ 1 & l = 0 \end{cases}. \quad (14)$$

Fig. 4 shows the parallel implementation alternative to the 2D WT.

Apart from preserving the compact support and reducing the complexity of the backprojection operation, 2D WT can be employed to obtain a multiscale representation of the projection data which, if implemented in parallel, can speed up the reconstruction process dramatically.

B. *Fast parallel algorithm*

Under the concepts exposed in the previous sections, by exploiting the attributes provided by the 2D WT when being employed in the image reconstruction process, we formulate a fast parallel algorithm. The main motivation in using the 2D WT is to take advantage of its time-frequency localization properties and be able to reconstruct accurately, independently

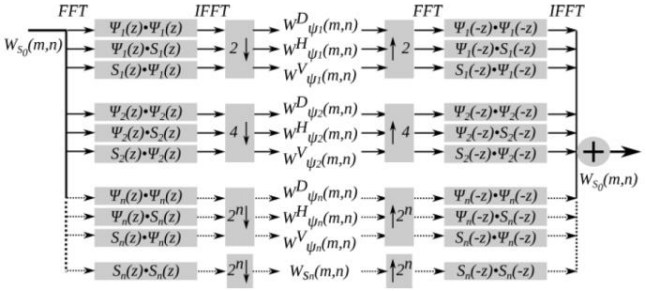


Fig. 4. Parallel two-dimensional separable wavelet transform.

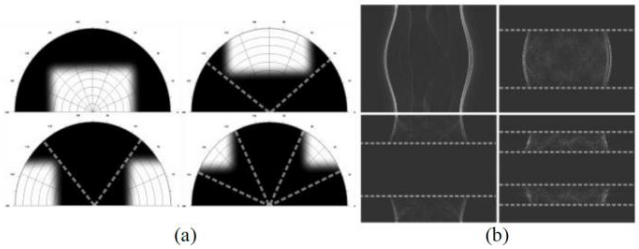


Fig. 5. Tomography projection data wavelet filtering. (a) 2D wavelet filters in Fourier domain and polar coordinates (0 to 180º), constructed by using the discrete Meyer FIR filter. (b) Wavelet-filtered projection data.

and in parallel, reduced area block components tiling together a full-size image. The main objective behind the reconstruction of block components is to reduce the overall reconstruction time, by considering only projection data corresponding to one block at a time.

A secondary feature, derived from the wavelet filtering of projection data and the sampling theorem, allows backprojection of wavelet coefficient images to be performed without loss of information onto subsampled grids, half the size of the complete resolution image obtained through standard FBP reconstruction,. This feature has an important implication in the speed-up of the reconstruction process, given the fact that backprojection is the most computationally expensive operation within the FBP [9].

Following the same line, by analyzing in Fig. 3 the breakdown of projection data within the Fourier space after wavelet filtering, it is obvious that certain projection angles have a null contribution (see Fig. 5) in the generation of detail coefficient images [11] and can be discarded without compromising the quality of the wavelet coefficient images. Conversely, when it comes to the generation of the approximations coefficient image, all projection angles must be considered (see top left example in Fig. 5(a)), therefore this advantage is only partially usable.

With the purpose to alleviate the consequences from such a restriction, the parallel multiscale decomposition scheme was adopted in the development of this algorithm. As explained before, this scheme allows a frequential decomposition of projection data, in which approximations coefficient is only computed for the lower resolution component. This means that, in relationship with the sampling theorem, approximations coefficient can be backprojected onto a less dense grid, which is reduced by a factor of two at every step down in resolution.

The algorithm presented in this work is a generalization of the FBP that has been designed by using MATLAB as software testing framework, having in mind a parallel implementation, either in hardware or in a software platform, capable of independently executing every component; e.g. in Simulink.

C. *Computer Implementation of the standard FBP algorithm*

In order to be able to adapt the standard FBP to the purposes of this work, it was implemented in MATLAB using the '*iradon*' function and also used as a reference benchmark.

The ramp filter design uses a method reported in [27] and [28], aiming to suppress the dc shift and inter-period interference artifacts. This is achieved by calculating an impulse response approximation of the ramp filter and matching the sampling interval to the bandwidths of the filter and the projection data, which are also zero-padded before the FFT is applied. Filtering is performed in the Fourier domain, and with the objective of reducing the algorithm complexity only "positive" frequencies are processed. The already filtered projection data is complemented with its complex conjugate, and IFFT is calculated. The backprojection operator, along with linear interpolation, is performed as designed in the '*iradon*' function.

D. The multiscale parallel block reconstruction algorithm

As previously mentioned, the algorithm was formulated for a parallel implementation. The latter depends on the isolation between certain elements, chosen among those that do not rely on values obtained during the execution of the algorithm and can be calculated a priori, and those that are totally dependent.

1) Off-line algorithm elements

The computation of the off-line elements starts from the decision of the scale depth into which the projection data will be decomposed. This information is relevant for the calculation of wavelet and scaling functions, as well as to determine the size of the wavelet coefficient images at each scale. In the algorithm, this information also determines the area of the blocks in which the full-size image will be decomposed, which is the area of the lower scale coefficient image, e.g. for a 256×256 pixel image, if decomposed into l = 3 scales, the lower scale image area is given by 256×2^{-3}, which corresponds to a 32×32 pixel image. This means that the full-scale 256×256 pixel image will be divided into four 32×32 pixel coefficient images (one approximations and three details) at scale l = 3, three 64×64 pixel detail images at scale l = 2, divided into four 32×32 blocks, and three 128×128 pixel detail images at scale l = 1 composed by sixteen 32×32 blocks. This is illustrated in Fig. 6.

By having information about the resolution depth and area size of each of the wavelet coefficients images, as well as of the constituent blocks, it is possible to calculate the filter coefficients (scaling/wavelet functions) by which the projection data will be decomposed. For this operation, (13) and (14) are employed in order to obtain the filter coefficients corresponding to each scale. These filter coefficients along with the designed ramp filter, are used to create the wavelet modified filters, directly employed in the analytical expression given by (11).

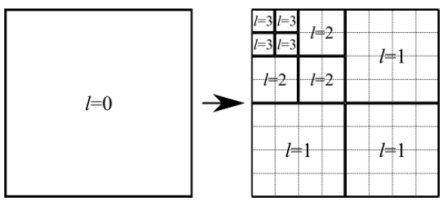


Fig. 6. Multiscale constituent block-breakdown

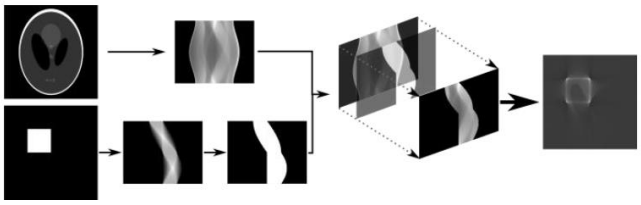


Fig. 7. Procedure of the projection data block-decomposition.

In order to decompose the projection data into several constituent blocks, a simple procedure was developed. It consists of generating synthetic phantoms representing the block area, within the field of view, that will be reconstructed individually. The RT from the block phantoms is then calculated and its support is obtained and converted to binary pixel values. The result is an array that is used as a template, to extract the projection data relevant for the reconstruction of the image block. This process is illustrated in Fig.7.

This approach has been developed with the purpose of avoiding the difficulties involved in the collection of off-centered data without degrading considerably the reconstructed image. In contrast to what is proposed in [21], this approach does not need extra padding of the projection data array, and therefore avoids the extra computational time involved when applying the filter.

2) On-line algorithm elements

The on-line elements are sequential portions of the algorithm that result as consequence of decomposing the reconstruction task into smaller ones. Each component task incorporates the computation that concerns only to the reconstruction of a wavelet coefficient block-tile image, at its corresponding scale, and is limited by its minimum data calculation dependency. Off-line calculated variables are required in the execution of every block.

Each component task has the same base operators of the standard FBP: filtering of the projection data in the Fourier domain and its backprojection onto the real space grid. The difference lies in that every component task is dedicated to the reconstruction of a reduced area, at every scale, which is possible because of the 2D WT incorporation. For the effect of dividing the full- size image reconstruction into a set of constituent block images, be reflected in speed gain, every component must deliver accurate images when provided only with data corresponding to the block area of interest.

Fig. 8 illustrates the processing carried out by each component task, when 3-level scale decomposition is desired. The first stage is the block decomposition of projection data, performed with the aid of support templates, for every scale except for the lower one, which determines the size and number of blocks in which the projection data is decomposed. FFT is then individually applied to the decomposed projection data and wavelet-based ramp-filtering is performed in the Fourier domain, resulting in the frequential decomposition of every projection data component. Further, IFFT of the filtered projection data components is taken, followed by the backprojection onto subsampled grids to create wavelet coefficient block-images at each scale. Finally, wavelet

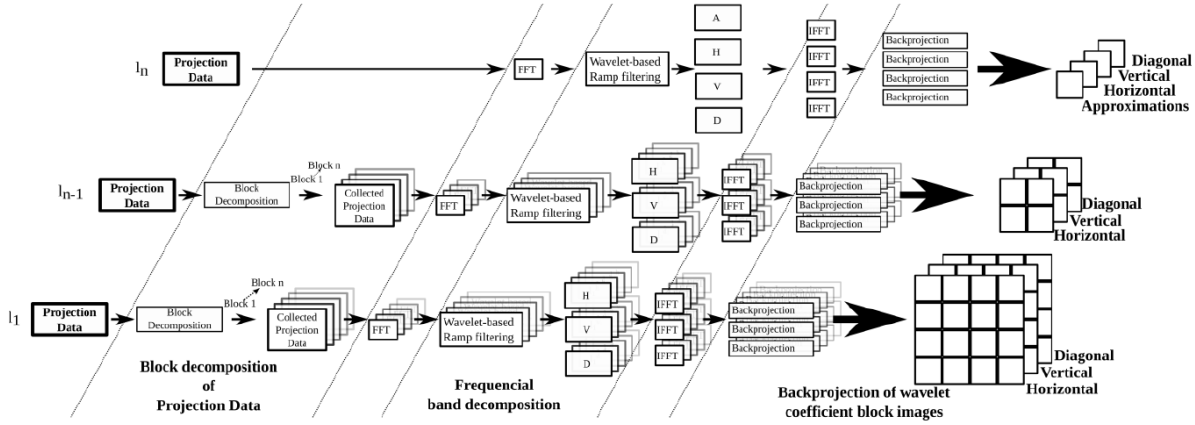


Fig. 8. Multiscale parallel block reconstruction of wavelet coefficient images

coefficient images are obtained from the tiling of block-images. The full-size image at maximum resolution is obtained by the merger of all wavelet coefficient images through inverse 2D WT in the space domain.

IV. RESULTS

Results shown in this section were obtained with synthetic forward RT data generated by the MATLAB '*radon*' function, from 180 view angles equispaced within [0,180º] on the standard Shepp-Logan test image. FBP image reconstruction was performed on a 512×512 pixel grid, no window function was employed in the filtering stage and linear interpolation was involved in backprojection operator. For frequential decomposition of projection data, a five-step multiscale representation was obtained through the parallel separable 2D WT. Wavelet coefficient images at each scale, except for scale l = 5, are divided into a set of 16×16 pixel constituent blocks; 4 for l = 4, 16 for l = 3, 64 for l = 2, and 256 for l = 1. The mother wavelet used for final results, was the symmetric and compactly supported "biorthogonal 2.4", which is length-10 and contains five zero moments. The wavelet coefficient images, as well as the synthesized full-scale image, are shown in Fig.9. Fig. 10 shows the output of the three variants of the FBP algorithm: standard FBP, 5-step multiscale FBP and parallel multiscale FBP with block tiling.

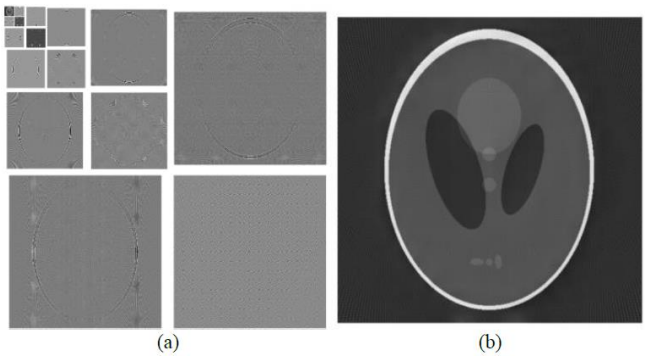


Fig. 9. Reconstructed image from the wavelet-based multiscale parallel block reconstruction algorithm. (a) Wavelet coefficients' images obtained at each scale; detail coefficient images for l=1,2,3,4,5 and approximations coefficient image for l=5. (b) Reconstructed output image at full scale l=0.

With the objective to study the speed gain achieved in the parallel implementation compared to the standard FBP image reconstruction, time analysis of the algorithm execution was carried out. Measurements were taken by using the 'tic/toc' function in MATLAB 2012b, with a 3.1GHz Intel Quad Core i3 processor and 4GB RAM computer system, as well as a 64bit Linux Fedora 17 operating system. To ensure consistency, measurements were repeated until reaching uniformity, and a single computer processor core was employed.

In the first instance, time measurements were taken from the multiscale wavelet-based FBP, with the objective of showing the speed gain from the inclusion of the 2D WT, without block decomposition. Fig. 11 shows the execution time, at each scale and for every wavelet coefficient image, calculated by the multiscale wavelet-based FBP. The time performance shown includes up-sampling and reverse-filtering. All execution times were considerably less than for the standard FBP, which under the circumstances was measured to be 1.7s.

For the parallel block decomposition algorithm, time performance was measured for each constituent block, for every wavelet coefficient, and at each scale. A detailed visual representation is impossible for this set of time performance measurements, as it would constitute a 512×512 pixel grid divided into 1024 16×16 pixel blocks. Therefore Fig. 12 presents the longest reconstruction time taken for a block within a specific scale and wavelet coefficient image. Again, up-sampling and reverse-filtering are included. For the

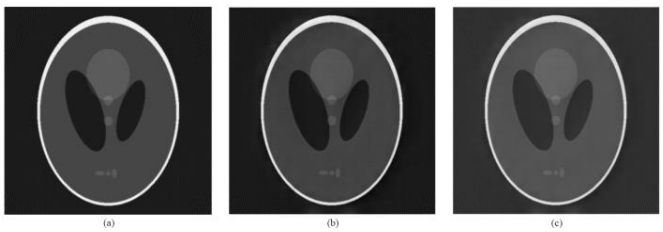


Fig. 10. Comparison between reconstructed images from different algorithms. (a) Standard FBP. (b) Wavelet-based multiscale five-step FBP. (c) Wavelet-based multiscale five-step parallel block FBP.
Full-size images of Fig 10 (a) (b) and (c) are provided as separate media.

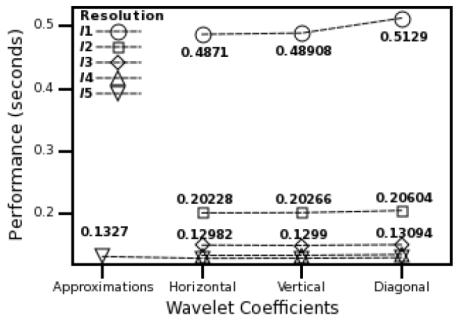


Fig. 11. Time performance of wavelet-based multiscale five-step FBP.

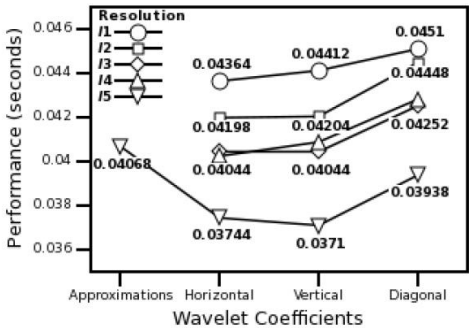


Fig. 12. Time performance of wavelet-based multiscale five-step parallel block FBP.

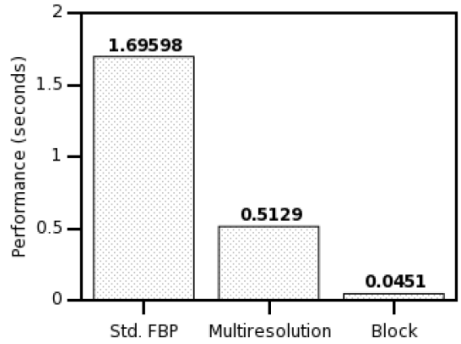


Fig. 13. Time performance between the three different reconstruction algorithms; Standard FBP, the wavelet-based multiscale five-step FBP and the wavelet-based multiscale five-step parallel block FBP.

purpose of comparison, the longest time manifested in Fig. 12, 0.0451 s, is taken as the total time spent on the execution of the parallel algorithm. Fig. 13 shows the final time performance comparison between the three algorithms.

V. Discussion

In this paper, an alternative algorithm to the widely used FBP has been presented. Due to its parallel implementation and block reconstruction approach, the image reconstruction speed is around 36 times faster than its standard counterpart;. This gain speed performance it is obtained at the expense of a slight image degradation (see Fig. 10), which is acceptable for many applications of FBP.

The work on the algorithm design demonstrated that the achievement of gain in speed at minimal image quality degradation, strongly depended on the type of wavelet transform to be employed and the choice of mother wavelet function.

The preferred choice of transform was the separable 2D WT, mainly because it is amenable to a parallel implementation in addition to delivering a multiscale representation. In the first instance, the importance of employing a wavelet filter that allowed an exact reconstruction scheme, while selecting the proper wavelet basis, was recognized. Attempts using different wavelet basis showed that, given the nature of the problem, a basis capable of preserving linear phase was required, which resulted in the choice of the biorthogonal wavelets [29].

The choice in terms of wavelet filters was subject to a more complex process. This was driven mainly by a priori knowledge, derived from the understanding of the tomography problem, as well as from the test results during the algorithm design. An indicator of the localization in Fourier domain is regularity and is determined by the wavelet filter smoothness. Higher regularity allows more accurate frequential decomposition, but at the expense of higher amounts of filter coefficients, resulting again in speed loss.

After managing to achieve exact inversion of the 2D WT, the major concern was to evaluate biorthogonal wavelet filters constructed with a different number of zero moments, supports and number of coefficients. The number of zero moments had a direct implication in the block-reconstruction accuracy, given the fact that it represents an indicator of the localization in space and is proportional to the support, which means that the higher the space localization, the less compact the support and the higher demand on computational resources. In the suggested algorithm, this was accounted for when deciding the amount of data to be collected for the reconstruction of constituent blocks: higher zero moments wavelets delivered higher quality reconstructed block images, at the expense of more data processing needed, resulting in less speed gain.

In conclusion, we have shown that tiled-block image reconstruction by wavelet-based, parallel filtered back-projection leads to more than an order of magnitude gain in speed, compared to standard FBP. This motivates future work in embedding such algorithms in programmable hardware, such as FPGAs.

References

[1] J. Radon, “On the determination of functions from their integral values along certain manifolds,” IEEE Trans. Med. Imaging, , vol. 5, no. 4, pp. 170–176, 1986.

[2] G. Hounsfield, “Apparatus for examining a body by radiation such as X or gamma radiation,” US Pat. 3,944,833, 1976.

[3] M. Holschneider, “Inverse Radon transforms through inverse wavelet transforms,” Inverse Probl., vol. 7, no. 6 , pp. 853-863, 1991.

[4] F. Rashid-Farrokhi, K. R. Liu, C. a Berenstein, and D. Walnut, "Wavelet-based multiresolution local tomography.," IEEE Trans. Image Process., vol. 6, no. 10, pp. 1412–1430, Jan. 1997.
[5] M. Costin, D. Lazaro-Ponthus, S. Legoupil, P. Duvauchelle, and V. Kaftandjian, "A multiresolution image reconstruction method in X-ray microCT," 2009 IEEE Nucl. Sci. Symp. Conf. Rec., M13-309, pp 3871-3876, 2009.
[6] L. Li, H. Toda, T. Ohgaki, M. Kobayashi, T. Kobayashi, K. Uesugi, and Y. Suzuki, "Wavelet-based local region-of-interest reconstruction for synchrotron radiation x-ray microtomography," J. Appl. Phys., vol. 102, no. 11, p. 114908, 2007.
[7] X. Yin, B. Ng, and J. Zeitler, "Local computed tomography using a THz quantum cascade laser," IEEE Sensors J, vol. 10, no. 11, pp.1718–1731, Nov. 2010.
[8] K. Niinimäki, S. Siltanen, and V. Kolehmainen, "Bayesian multiresolution method for local X-ray tomography in dental radiology," Proc SPIE, vol. 7246, p. 72460A–11, Feb. 2009.
[9] A. H. Delaney and Y. Bresler, "Multiresolution tomographic reconstruction using wavelets.," IEEE Trans. Image Process., vol. 4, no. 6, pp. 799–813, Jan. 1995.
[10] T. Rodet, L. Desbat, and P. Grangeat, "Parallel algorithm based on a frequential decomposition for dynamic 3d computed tomography", Proc. Int. Parallel Distrib. Proces. Symp., pp 7-14, 2003.
[11] T. Rodet, P. Grangeat, and L. Desbat, "Multichannel algorithm for fast 3D reconstruction.," Phys. Med. Biol., vol. 47, no. 15, pp. 2659– 71, Aug. 2002.
[12] F. Natterer and G. Wang, "The Mathematics of Computerized Tomography," Med. Phys., vol. 29. p. 107, 2002.
[13] K. J. Batenburg and L. Plantagie, "Fast approximation of algebraic reconstruction methods for tomography.," IEEE Trans. Image Process., vol. 21, no. 8, pp. 3648–58, Aug. 2012.
[14] K. Ogawa, M. Nakajima, and S. Yuta, "A reconstruction algorithm from truncated projections,", IEEE Trans. Med. Imaging, vol. 3, pp.34-40, March 1984.
[15] A. Louis, "Incomplete data problems in X-ray computerized tomography," Numer. Math., vol. 262, pp. 251–262, 1986.
[16] A. Faridani, "Local tomography," SIAM J. Appl. Math., vol. 52, no. 2, pp. 459–484, 1992.
[17] P. Maass, "The interior Radon transform," SIAM J. Appl. Math., vol. 52, no. 3, pp. 710–724, 1992.
[18] A. Bilgot, L. Desbat, and V. Perrier, "Filtered backprojection method and the interior problem in 2D tomography," Nucl. Sci. Symp. Med. Imaging Conf (NSS/MIC) 2009, pp. 4080-4085, 2011.
[19] G. Wang, H. Yu, and B. De Man, "An outlook on x-ray CT research and development," Med. Phys., vol. 35, no. 3, p. 1051, 2008.
[20] G. Wang and H. Yu, "The meaning of interior tomography," Phys. Med. Biol., pp. 5764–5767, 2013.
[21] T. Olson and J. DeStefano, "Wavelet localization of the Radon transform," IEEE Trans. Sig. Process., vol. 42, no. 8, pp. 2055–2067, 1994.
[22] T. Olson, "Optimal time-frequency projections for localized tomography.," Ann. Biomed. Eng., vol. 23, no. 5, pp. 622–36, 1995.
[23] C. A. Berenstein and D. F. Walnut, "Wavelets and Local Tomography," Wavelets in Medicine and Biology, pp 231-261, CRC Press, pp 231-263, 1996.
[24] I. Daubechies, Ten Lectures on Wavelets (CBMS-NSF Regional Conference Series in Applied Mathematics). 1992, p. 377.
[25] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," IEEE Trans Pattern Anal. Mach. Intell., vol. 11, no. 7, 674-693, 1989.
[26] F. Tay and J. P. Havlicek, "Frequency implementation of discrete wavelet," 6th IEEE Southwest Symp. Image Anal. Interpret., 2004. pp. 167–171, 2004.
[27] A. C. Kak and M. Slaney, Principles of Computerized Tomographic Imaging. IEEE Press 1988, p. 344.
[28] A. Kak, "Computerized tomography with X-ray, emission, and ultrasound sources," Proc. IEEE, vol. 67, no. 9, pp. 1245 – 1272, 1979.
[29] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform.," IEEE Trans. Image Process., vol. 1, no. 2, pp. 205–220, Jan. 1992.

Jorge Guevara Escobedo obtained his BEng in Electronics and Telecommunication Engineering and MSc in Electronics from the Autonomous University of the State of Hidalgo. Hidalgo, Mexico.(2006) and the National Institute of Astrophysics, Optics and Electronics, Puebla, Mexico (2007). He gained industrial experience as Design Engineer in Puebla, before receiving in 2011 a Doctoral Studentship award from Consejo Nacional de Ciencia y Tecnología (CONACyT), Mexico. Currently he is Doctoral Candidate and a Postgraduate Teaching Assistant at the University of Manchester, UK. His main areas of interest are signal processing for imaging sensors, embedded signal processing and imaging.

Krikor B. Ozanyan (M'1995, SM'2003) received his MSc degree in engineering physics (semiconductors) and PhD degree in solid-state physics, in 1980 and 1989 respectively, from the University of Sofia, Bulgaria.

He has held previous academic and research posts in the University of Sofia, The Norwegian Institute of Technology (Trondheim, Norway), the University of Hull (UK), and the University of Sheffield (UK), working on projects ranging from Brewster-angle mid-IR spectroscopic ellipsometry and electron confinement in quantum wells and barriers, to the demonstration of the lasing at 333nm from strained MQW ZnCdS/ZnS structures and *in-situ* real-time optical monitoring of growth of III-V semiconductors in MBE and MOCVD reactors. His current interests are in the area of photonic sensors and indirect imaging (tomography) by optical modalities, signal processing for optical experiments, and spectroscopy with ultrafast laser sources. He is currently Head of Sensors, Imaging and Signal Processing at the University of Manchester and Visiting Professor at the University of Bergen, Norway.

Professor Ozanyan is Fellow of the Institute of Engineering and Technology, (UK, formerly IEE) and Fellow of the Institute of Physics (UK). He was Distinguished Lecturer of the IEEE Sensors Council in 2009-2010, Guest Editor of the IEEE Sensors Journal Special Issues "Sensors for Industrial Process Tomography" in 2005 and "THz Sensing: Materials, Devices and Systems" in 2012. He is currently Editor-in-Chief of the IEEE Sensors Journal.