5-2010

Development and Application of Computational Tools for the Study and Optimization of Variable Resolution X-ray (VRX) Computed Tomography Scanners

David Alejandro Rendon
University of Tennessee Health Science Center

Development and Application of Computational Tools for the Study and Optimization of Variable Resolution X-ray (VRX) Computed Tomography Scanners

DOI
10.21007/etd.cghs.2010.0258

Development and Application of Computational Tools for the Study and Optimization of Variable Resolution X-ray (VRX) Computed Tomography Scanners

A Dissertation
Presented for
The Graduate Studies Council
The University of Tennessee
Health Science Center

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy
In the Joint Graduate Program in Biomedical Engineering and Imaging
From The University of Tennessee
and
The University of Memphis

By
David Alejandro Rendon

May 2010

To those who never stopped believing in me
and always supported me with infinite patience
My mother Myriam
My father Arnul
My brother Dario
My mentor Frank

Acknowledgements

I would like to thank all those who in one way or another helped me throughout the process of taking this project to fruition. I am most thankful to Dr. Frank DiBianca, because this work would have never been completed without his excellent technical knowledge, his extraordinary role as a mentor and friend, his unfaltering support even in the toughest times, and the incredible amount of patience he displayed. I am also very grateful to Dr. Gary Keyes for his seemingly infinite help and support were fundamental for my success.

There are many other people whose help allowed me to advance through different parts of this project. In particular, I would like to thank Dr. Lawrence Jordan for his fundamental help in the early stages, as well as Joseph Laughter and Dr. Roman Melnyk for their frequent help and advice, and Dr. William Veeser for his advice in the late stages. Finally, I would also like to thank all the lab mates who were with us throughout these years, and in particular Dr. Bahram Dahi for his friendship and assistance both in and out of the lab.

This study was partially supported by the NIH Grant No. EB-00418.

Abstract

The overall goal of this project was to develop and apply important computerized aids for the design and implementation of Variable Resolution X-ray (VRX) CT scanners developed at the University of Tennessee, Memphis. VRX scanners take advantage of the "projective compression" principle that allows the same device to image objects of different sizes with the same level of detail by adjusting the field of view and the reconstruction resolution.

The first part of this project aimed to develop a set of computational tools specifically tailored for the design, implementation and study of VRX scanners. This included creating a reconstruction algorithm that takes into account the unique geometries of the different VRX systems that have been designed, along with improving the calibration algorithm needed to ensure a proper reconstruction. It also included the development of a computer model of VRX scanners that is an invaluable tool for the development and study of these devices.

The second part of the project was composed of a small series of experiments in which the computational tools developed proved to be fundamental in the analysis and evaluation of some aspects of VRX imaging. This included a comparison of the performance of different targeting VRX geometries in terms of spatial and contrast resolution, and a study of the effect of the VRX angle on the severity of common artifacts in single-arm images.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CAT, CT	(X-ray projection) Computed (Axial) Tomography
CBCT	Cone Beam Computed Tomography
ESF	Edge Spread Function
FBP	Filtered BackProjection
FDK	Feldkamp reconstruction algorithm
FFT	Fast Fourier Transform
FOV	Field Of View
keV	Kilo Electron Volt
kVp	Peak Kilovoltage
LSF	Line Spread Function
MTF	Modulation Transfer Function
OTF	Optical Transfer Function
PSF	Point Spread Function
ROI	Region of Interest
SNR	Signal to Noise Ratio
VRX / VRX CT	Variable Resolution X-ray Computed Tomography

Chapter 1

Introduction

Computed tomography (CT) is a medical imaging technique in which the composition of cross sections of the patient is determined with the use of x-ray devices. For this, an x-ray tube and x-ray detectors are used to obtain a large number of projections from many different angles in the plane of the cross section. Each projection carries information of x-rays that have been attenuated as they traversed the patient, with the degree of attenuation corresponding to the composition and size of the objects in the ray's path. These projections are processed by a reconstruction algorithm which produces the tomographic slice, which is in principle a map of the linear attenuation coefficients of the corresponding regions in the patient's anatomy. Since the different tissues will normally have different attenuation coefficients, this map serves as a picture of the internal organs of the patient, reflecting with detail their current structures and many anomalies. As such, CT scanners have become one of the main tools in modern diagnostic imaging. The use of CT scanners has long transcended the clinical fields, as they are increasingly common in many fields of research and industry.

Commercial CT scanners are normally designed for a particular purpose: clinical scanners are designed for imaging human patients and therefore are large enough to accommodate most adults. On the other extreme, micro-CT scanners provide much higher spatial resolution, but are only large enough to hold small animals or samples. The resolution of these devices is, up to a certain extent, fixed. Thus, imaging a small animal (or, for that matter a small child), in a clinical scanner will constitute an under-utilization of the field of view (FOV) of the scanner, and the reconstructed images will not be as useful because the size of the structures that are being imaged is much smaller than those of full-sized patients. In other words, the resolution provided by the large scanner is insufficient or inadequate to image the small subjects satisfactorily. As such, it would be ideal to have several scanners with different FOVs and corresponding resolutions, and use the one that better matches each patient's size. Since such a scenario is economically unreasonable, it would be desirable to have a scanner that adjusts its resolution to the size if the patient, reducing the FOV to match that size if necessary.

Variable Resolution X-ray (VRX) CT scanners attempt to address this problem by taking advantage of the *projective compression* principle [17, 14], i.e., by making use of the fact that the projected size of an x-ray detector cell will decrease as the detector is tilted, thereby allowing it to register a higher spatial resolution at the

expense of having a smaller FOV. The most basic VRX scanners have a single linear detector array, or *arm*. More sophisticated versions use two or four arms, allowing a better use of the space as well as the possibility of target imaging (see Section 3.1.3). Along with the reduction of the FOV, the application of the VRX concept has other drawbacks. One of them is that there may be some cross-talk between detector cells (photons that are not captured by the intended cell may traverse it and be detected by a neighboring cell) and between arms (caused by back-scattering). Nevertheless, it has been shown that the negative effect of the crosstalk due to cell traversal is normally smaller than the gains in spatial resolution due to the VRX effect, because the amount of x-rays photons is reduced exponentially as they traverse the detector cells [41]. Another potential drawback of applying the VRX concept is that at extremely small VRX angles a considerable number of photons may be reflected off the arm surface; fortunately this only occurs at "sub-microscopic" angles and can be overcome with a stair-step cell design that is still under development [14]. A third drawback is that proper calibration of multi-arm systems becomes a delicate issue.

The design and optimization of VRX scanners demands the development of special tools. Reconstruction and calibration algorithms designed for traditional scanners must be adapted due to the unique geometry of the detectors used. Also, other tools of fundamental value for the study of CT scanners such as simulators must be designed from the ground up as the tools currently available are either not applicable to VRX research or their practical implementation makes their use unfeasible.

The main goal of this project is to provide several computational tools needed for multi-arm VRX research and employ those tools in various studies, including the effect of VRX principles on some general aspects of CT imaging and comparisons of different scanner implementations and configurations. To achieve these goals, the project is divided in two major phases. The first phase is the development of computer software tools that support research. In the second phase, these tools are used for the analysis and evaluation of VRX CT systems and configurations under consideration.

1.1 Development of Computational Tools

The first major goal of the project, the development of computational tools for VRX scanners, is addressed through two specific aims: the adaptation and optimization of general CT algorithms (e.g., reconstruction and calibration) for multi-arm VRX scanners, and the development of computational models of these scanners as an aid for research.

1.1.1 Development and Optimization of Algorithms for VRX CT Scanners

The unique architecture of VRX scanners demands the modification of the CT reconstruction algorithms. This aim includes the development of a reconstruction algorithm that takes special consideration for the tilt of the scanner arms and support for an arbitrary number of arms in various geometries (including four-arm systems for target imaging), as well as other peculiarities of these type of scanners, as described in Section 2.3. It also includes the development of supporting algorithms such as geometrical calibration of the device.

Chapter 2 provides an introduction to VRX scanners in the context of general CT imaging, and the specific algorithms developed for them will be discussed in Chapter 3. The code for the VRX CT reconstruction and calibration algorithms is presented in Appendix A.

1.1.2 Development of Computer Simulators of VRX Scanners

Computer modeling is an invaluable tool in the development, optimization and evaluation of complex devices such as imaging instruments. Unfortunately, the tools available for simulation of traditional CT scanners and other x-ray devices are not adequate for the study of VRX scanners. Some tools are too limited to allow the modeling of a VRX configuration, while others are too complex and computationally demanding to be useful with the available resources. Therefore, the need for the in-house development of VRX simulators has become apparent.

The specific goal of this part of the project is the design and implementation of a computer simulator that incorporates all the distinctive aspects of VRX scanners, while modeling only the physical phenomena that we consider most significant, relevant to our research, and that can be handled with our computational resources. It is important to note that many of the concepts developed in this process are nevertheless also applicable to other types of CT scanners and x-ray imaging devices, so their potential value far exceeds the realm of VRX imaging.

The simulators developed for this project are discussed in Chapter 4. The discussion includes a description of the physical phenomena that is modeled and how they are implemented. It also includes the validation procedures followed to guarantee that the results obtained through the simulator are suitable representations of the real devices that are being modeled. The code of the VRX computer simulator and other supporting programs is listed in Appendix B.

1.2 Analysis and Evaluation of VRX Systems and Configurations

The second major goal of this project is the analysis and evaluation of VRX systems and configurations through the use of the tools developed in the first phase. This goal will be addressed through two specific studies. The first study will show the enormous value of the computational tools for the study of different CT scanner implementations and configurations. In the second study, the effect of the projective compression principle on different aspects of CT imaging is studied. It is important to note that, in addition to showcasing the power of the computational tools, both of these studies are by themselves interesting contributions to our understanding of VRX scanners and thus of CT scanners with unusual detector geometries.

1.2.1 Comparison of Different VRX CT Implementations and Configurations

A VRX system can be *implemented* in many different ways depending on the detector technologies used, the x-ray tube specifications, the image acquisition parameters, etc. Given a specific set of materials, a VRX system can also be *configured* in many different ways. For example, the detectors in a four-arm system can be coupled to form a two- or even a single-arm system, and in multi-arm systems the arms can be placed in different positions to allow for target imaging. In target imaging, a region at the center of the image, known as the *target zone*, is imaged at a higher resolution while the external field of view remains large [13]. Target imaging with multi-arm VRX scanners is discussed in Section 3.1.3.

In Chapter 5 we assess how different choices for the implementation and configuration of a targeting VRX scanner will affect its performance. For this, a series of virtual experiments were designed for the comparison of different VRX implementations and configurations in terms of spatial and contrast resolution. These results are useful for researchers that wish to determine which prototypes and configurations should be preferred without the huge investment of time and money required to test each prototype physically.

The study examined in Chapter 5 was focused on three targeting VRX scanners that share many characteristics, the main difference being the way the detector arms were configured to produce the target region.

1.2.2 Study of the Effect of Projective Compression on Different Aspects of CT Imaging

It has been proven [17, 14] that due to the projective compression principle the spatial resolution of VRX scanners will in general increase as the VRX angle

decreases. Nevertheless, there has not been very extensive work on the influence of projective compression on other aspects of CT imaging, such as contrast and contrast resolution, the production and severity of artifacts, and the effects of miscalibration.

Chapter 6 describes a study that explored how changing the VRX angle in a single-arm scanner affects the severity of some types of artifacts typically found in CT images, including streaks caused by thin objects of highly absorbent materials and beam hardening-related artifacts such as shadowing produced between radiopaque objects.

1.3 Other Potential Applications of the Computational Tools

The studies described in Chapters 5 and 6 show the potential of the computational tools developed for this project and described in Chapters 3 and 4. Nevertheless, the possible applications of these tools are far more reaching. In fact, some of the concepts studied and several of the techniques developed for the creation of these tools can be easily extrapolated to non-VRX geometries for CT scanners, and even to other types of x-ray based devices. Chapter 7 explores some of these potential applications and summarizes the goals achieved by this project.

Chapter 2

Variable Resolution X-ray (VRX) CT Scanners

In the medical imaging field, the term Tomography refers to several imaging modalities in which the internal organs of the patient are visualized through slices or cross-sections. Tomography modalities include x-ray transmission computerized (or computed) axial tomography (CAT or, more commonly, CT) scanners, PET (positron emission tomography), SPECT (single photon emission computerized tomography), ultrasound, and MRI (magnetic resonance imaging).

In this work we are interested specifically in x-ray transmission CT scanners, in which hundreds of x-ray images acquired from different angles around the patient are used to compute the tomographic slice. X-ray transmission CT is one of the first and most widely used tomography modalities; for this reason, it has traditionally been referred to simply as Computed Tomography or CT. In this modality, the internal configuration of the patient's organs is visualized through the use of hundreds of projection images obtained with an x-ray source and detectors. The tomographic slices are computed mathematically by combining information from these projection images, hence the name "Computed Tomography". The process through which the slices are calculated is called CT reconstruction.

In this work we deal with a specific type of CT called Variable Resolution X-Ray Computed Tomography, or VRX CT. Before dwelling on VRX CT scanners, we will present a brief description of the theory and evolution of modern CT scanners in general. In Section 2.1.1 we will talk about the different generations of clinical CT scanners, several of which are still in use, and in Sections 2.1.2 and 2.1.3 we will extend the overview to other CT technologies that depart from the generations of clinical scanners described in Section 2.1.1 even though they are all closely related. Some algorithms associated with computed tomography, including reconstruction algorithms and algorithms used to calibrate the scanners will be discussed in Section 2.1.4. In Section 2.2 we describe some parameters used to evaluate the quality of the images produced by the reconstruction process. Finally, in Section 2.3 we will introduce different types of VRX CT scanners, including the ones studied throughout the remaining chapters.

2.1 Traditional Computer Tomography Scanners

All x-ray transmission CT systems produce tomographic slices by combining information from hundreds of radiographic images obtained from different angles around the patient or object being scanned. For this, the scanner is equipped with an x-ray source, x-ray detectors and a mechanism that will allow the acquisition of the projection images from the different angles. Throughout the history of CT scanners there has been a constant evolution of all these parts as well as the procedures and algorithms involved, allowing the generation of images of higher quality and diagnostic value in shorter times. In this section we will discuss some of the most significant CT scanner configurations that have been used historically to provide a backdrop for the VRX CT scanners that will be described in Section 2.3.

Although the real development of CT scanners began in the late 1960s and early 1970s with the introduction of digital computers powerful enough to handle the complex mathematical operations involved in CT reconstruction, the mathematical principles behind this procedure were first described by J. Radon in 1917 [8, 32]. Radon demonstrated theoretically that a cross-sectional image of the interior of a two- or more dimensional object can be reconstructed mathematically from the set of all the projections of the object in all directions. The procedure of obtaining this infinite set of projections is aptly called the Radon Transform, and the inverse operation that reconstructs a tomographic image from this infinite set is called the Inverse Radon Transform, which is the cornerstone of computed tomography [8, 32].

Practical applications of these principles were nevertheless unfeasible due to the large amount of calculations involved, so little progress was made in this area in the following decades. In 1957 A. M. Cormack applied Radon's mathematical underpinnings to the specific problem of x-ray computed tomography but the limitations still prevalent restricted the phantom to a single cylindrical object [32]. The first clinical CT scanner was introduced by G. N. Hounsfield in 1971. Due to physical constraints, that first scanner was specifically designed for imaging the brain [32].

In the last four decades constant advances in technology have led to a constant evolution of CT scanners. The most significant clinical CT scanner variations are traditionally grouped in five major generations [32], and some authors identify two additional generations [37]. We will overview all these generations in the next section, and in Section 2.1.2 we will extend the discussion to other important types of CT scanners.

2.1.1 Evolution of Clinical CT Scanners

The first CT scanner generation employed a pencil beam x-ray source and a single detector facing it. Both the source and the detector were shifted linearly in the direction perpendicular to their line of view to obtaining 160 data points corresponding to parallel beam x-ray projections traversing the patient. The source

and detector were also mounted on a gantry that would then rotate at 1° intervals. This operation would be repeated 180 times, thus acquiring views for half a rotation of the gantry. A complete scan was achieved in 4.5 minutes so patient motion was a major issue. Because of this and the fact that it had a fairly small field of view (24 cm), the system was limited mainly to brain scanning [24, 22, 32].

In the second generation the single detector was replaced by a linear array of up to 30 cells and the pencil beam source was replaced with a narrow fan beam (10°) that would span the detector array. Although the second generation scanners still operated in a translate-rotate mode, the use of multiple detectors reduced drastically the number of steps required in the rotate phase, shortening the scan time by a factor proportional to the number of cells in the linear array. This was due to the fact that at each step data were acquired not only by the cell directly facing the focal spot but also by the other cells in the array; those cells captured data corresponding to rays that transversed the patient through the same paths as the rays corresponding to those rotation angles that were skipped. Because of this reduction in the number of rotation steps, the scan time was reduced to 18 seconds even though in these scanners each slice was acquired by a full rotation of the gantry. The reduced scan time was short enough for most patients to hold their breath thus avoiding motion artifacts [32, 25, 37, 38].

The basic architecture of the third generation scanners is still the most used today. The translation phase is completely eliminated by employing a large number of detectors (frequently over 800) and a wide angle fan beam that covers the entire patient. The detectors are usually distributed on an arc centered at the x-ray tube. Because both are connected and there is no translational movement involved they can rotate around the patient at higher speeds, allowing scan times between 2 and 5 seconds in early models. This was later enhanced by replacing the power and data connections with slip rings. With the slip rings the gantry did not need to stop and rotate in the opposite direction to unwind the cables, reducing the scan time to under 0.5 seconds thus eliminating almost all motion artifacts for most typical applications [32]. One significant drawback of the third generation architecture is a high susceptibility to ring artifacts. This is due to the fact that each cell in the detector is the main contributor to the pixels located in a ring around the center of the reconstructed image. Therefore, if a particular cell is improperly calibrated or produces a signal with a particular (positive or negative) bias, the values of all the corresponding pixels will also be biased resulting in a bright or dark ring in the reconstructed image. This limitation has been significantly overcome with improved calibration and postprocessing algorithms [37, 43].

In order to avoid the ring artifacts of the third generation scanners, in the fourth generation the rotating detector array was replaced with a fixed 360° ring of detectors, so only the x-ray tube rotates. As a consequence, each cell of the detector will actually contribute to all pixels in the image, without the emphasis on a particular region exhibited by the third generation architecture. But in order to cover the full 360° ring, the number of detectors needed to be increased dramatically,

reaching up to 4800. This increase, along with the reduction of ring artifacts in the third generation discussed above, has made the implementation of fourth generation scanners impractical, especially for multislice machines [46, 32, 37, 49].

Although the scan time of modern third and fourth generation scanners is extremely good for most applications, it is still excessive for cardiac imaging. In some specialized scanners, this problem is solved by implementing cardiac gating, i.e., by synchronizing the exposure and collection of data with a specific part of the cardiac cycle registered by ECG [26, 12, 72, 65]. But before this technique became common, an entirely different approach was followed by the fifth generation or electron beam scanners which perform a scan in 50 ms (an adequate time for cardiac applications) by eliminating all mechanical motion. In these scanners, the tube is replaced by a high-energy electron beam source located behind the patient. The charged beam is directed in a sweeping motion along a tungsten arc shaped anode that surrounds the patient, generating a wide fan beam of x-ray photons. The signal produced after this x-ray beam traverses the patient is acquired by a ring of detectors surrounding the patient. As the electron beam sweeps around the tungsten arc the different projections are acquired in a way similar to a fourth generation scanner but in a much shorter time due to the absence of mechanical movement [8, 37, 32, 25].

Some authors consider classifications of a sixth and a seventh generation of CT scanners, both of which are actually direct descendants of the third generation [37]. In this sixth generation, or helical scanners, the patient is moved in the direction perpendicular to the gantry as the tube and detectors rotate several times. The information is therefore acquired in a helical path from which the projections corresponding to several slices can be inferred. In this way, a complete volumetric scan of the abdomen can be acquired in around 30 seconds, a time short enough for most patients to hold their breath. This allows the acquisition of multiple slices while avoiding motion induced artifacts constituting a great advance over step-and-shoot third generation scanners [8, 25, 37, 68].

Seventh generation scanners have several rows of detectors allowing the acquisition of multiple slices in a single rotation. Compared to single-slice helical CT scanners these multislice scanners offer an improvement in the slice thickness and the spatial resolution in the cross-plane direction (i.e., the direction of the rotation axis). Most modern clinical CT scanners are actually a hybrid of the sixth and seventh generations, that is they are multislice helical scanners that can scan large volumes in shorter times with fewer rotations of the gantry [68]. Multislice CT scanners also take better advantage of the x-ray tube by making use of photons directed out of the central plane that would otherwise be collimated out. The drawback is a higher level of noise due to scattered radiation, but that is reduced by the use of optimized collimators and improved detector design. These optimizations allow multislice helical scanners to produce images with up to 20% less noise than their single-slice helical counterparts [25]. Most commercial multislice CT scanners nowadays have between 4 and 64 detector arrays, with some state-of-the-art devices reaching 256 slices [8, 32, 37].

As mentioned before, most of the modern clinical CT scanners are multislice helical machines (sixth and seventh generation) and therefore use the basic architecture of the third generation as opposed to the fourth and fifth. Typical clinical CT scanners operate at a spatial resolution of 1.2 to 2.5 lp/mm in the tomographic plane with the slice thickness of up to 0.5 mm. A typical study, which ranges from a few centimeters to the whole body, is frequently composed of several hundreds and even thousands of individual slices, each one corresponding to a 512×512 pixel image. Such a study is acquired in 5 to 20 seconds. To achieve a contrast resolution of 3 HU, a radiation dose of 30 mGy is delivered [37].

There are other techniques that have been used to reduce the scan times of regular CT scanners making them more suitable for cardiac applications without further increasing the rotation speed of the gantry or resorting to the use of the extremely expensive hardware of the fifth generation. These techniques include the use of half-scan reconstruction algorithms that only require the information from projections for 180° around the patient , and dual-source CT in which two x-ray tubes are mounted on the same gantry at 90° from each other along with two corresponding sets of multislice detectors. By combining the information from both sets of x-ray tubes and detectors in a modified half-scan reconstruction algorithm, the information required to reconstruct a slice can be acquired with only 90° of rotation of the gantry. The implementation of such a hybrid device on a helical scanner can reduce the scan time to 83 ms even without the use of cardiac gating [1, 37]. Because the x-ray tubes can be operated at different voltages, dual source CT scanners are also used to provide improved tissue differentiation [33] by implementing dual-energy imaging techniques.

2.1.2 Cone Beam CT

In the previous section we presented an overview of the history and evolution of the most common kind of clinical CT scanner, i.e., the large gantry scanners specifically designed to allow imaging the torso of most adults. In addition to these scanners there are other classes of scanners designed for a large number of applications ranging from industry to animal research, and even clinical scanners that don't really fit into any of the generations described before. Of particular interest for this work are micro-CT scanners which will be discussed in Section 2.1.3, and cone beam scanners.

Cone beam CT scanners (CBCT) use relatively recently developed planar detector arrays, such as flat-panel detectors (FPD) or charge-coupled device (CCD) cameras. Rather than being composed of isolated detector cells these devices offer a two-dimensional array with hundreds of rows and columns of cells. In essence this is similar to the most advanced multislice (seventh generation) scanners discussed in Section 2.1.1 which have up to 256 rows of detectors, although CBCT systems have many more rows and far smaller cells which allow the acquisition of images of far higher resolution [7, 8, 64].

In CBCT imaging there is no translation of the subject and all the data necessary for the reconstruction of a 3-D volume is acquired in a single rotation. The reconstruction of such data sets requires special algorithms, the most common being the Feldkamp (FDK) algorithm which is a generalization to 3-D of the backprojection methods, that also takes into account the beam divergence in the transverse direction [21, 71]. Although other reconstruction algorithms have been developed including exact and iterative solutions, FDK is by far the most widely used due to its straightforward implementation and practicality [50].

Unfortunately due to the high cost of manufacturing of large planar detectors it is unfeasible to apply such a technology to the large whole body scanners, and the applications are presently limited to very small fields of view. In the clinical field CBCT is found mainly in dentistry applications where only the mouth and nearby structures are imaged but a higher resolution is required [44, 30], and in rough CT scans acquired with the rotating C-arms used in fluoroscopy [71]. The principles behind CBCT reconstruction have also been adapted in the reconstruction algorithms used by some helical multislice clinical scanners [66]. Nevertheless, the most common application of CBCT is in the micro-CT scanners described in the next section.

2.1.3 Micro-CT Scanners

Micro-CT scanners are miniature versions of computed tomography scanners used for imaging small animals and other small samples at very high resolutions. Since their introduction in the early 1980s, micro-CT scanners have become a fundamental and widespread tool for biomedical research, as they provide relatively cost-effective and minimally invasive methods to study the anatomy of animals used as models for many types of diseases. Although they have been used primarily for bone imaging due to the natural high contrast between osseous and soft tissue, micro-CT systems have also been successful in soft issue studies especially with the use of contrast agents [50, 54]. Micro-CT is also frequently used in conjunction with PET and SPECT, which makes it invaluable for many applications of molecular imaging such as in pharmacokinetics [70, 69].

Most micro-CT scanners can be divided in three general categories: scanners with rotating gantry which resemble a miniature clinical CT, scanners with rotating table, and the far less common synchrotron-based micro-CT scanners. Synchrotron-based scanners utilize a synchrotron monochromatic x-ray source. Although this kind of scanner can reach a spatial resolution of 1 µm or less allowing the imaging of subcellular structures, its availability is limited by its prohibitive cost [28, 54, 50, 55]. For the purpose of this work, we will focus our attention on regular micro-CT scanners that use an x-ray tube rather than a synchrotron x-ray source.

Scanners with rotating gantry are used mainly for in vivo imaging where the animal is kept in a fixed horizontal position throughout the scan while the gantry containing an x-ray tube and detectors rotates around it, in a way that resembles

most clinical CTs [50, 61, 54, 55]. In rotating table micro-CTs the subject to be imaged is placed on the center of a horizontal table that rotates at a constant speed, and the x-ray tube and the detectors are placed in fixed positions at opposite sides of it. Commercial rotating table micro-CT scanners are used mainly for in vitro studies, but due to the lower costs and the clear mechanical simplicity compared to a rotating gantry, the prototypes of scanners that explore new technologies (such as the VRX scanners studied here) are frequently built around a rotating table architecture [60, 61, 54].

Micro-CT scanners with rotating gantries are frequently used to image animals of sizes ranging from small mice to large rats and their field of view is usually between 5 and 10 cm. The typical spatial resolution ranges from 50 to 100 µm. The scan time will depend on many factors including the image quality desired but is usually less than 10 minutes. The parameters for rotating table scanners are very varied, but the ones used for studies of extremely small inanimate subjects such as biopsy sized specimens are optimized for higher spatial resolutions of around 10 to 50 µm. Their field of view is correspondingly much smaller, around 1.5 to 5 cm. Because the radiation dose is much less of an issue, the scan times are allowed to be much longer, typically from 10 to 30 minutes; this allows for the acquisition of images with far better contrast and perceived resolution [54, 31, 50, 22, 53].

Common micro-CT scanners employ microfocus x-ray tubes with focal spots smaller than 100 µm [54, 31]. They also use high resolution x-ray detectors with pixel spacing smaller than 50 µm [31, 54]. Modern micro-CT scanners usually employ CCD cameras or FPDs in a CBCT configuration [39, 54, 67, 59] (see Section 2.1.2). Due to the small size of the samples, the x-ray photons used in micro-CT imaging are frequently of a lower energy (around 25 keV) than those used in clinical scanners where energies in excess of 50 keV are the norm [60]. A consequence of this is that there will be more x-ray attenuation and it will be more dependent on the atomic numbers of the atoms in the sample allowing better tissue discrimination. On the other hand, the attenuation is more energy dependent and therefore there is more susceptibility to beam hardening artifacts. Because of this, the selection of the peak voltage of the tube is more critical as is the need to generate a closer to monochromatic beam (an ideal that in practice is only achieved by the synchrotron sources described above) [55, 53, 50].

2.1.4 Algorithms in Common CT Scanners

Unlike the different modalities of x-ray imaging based solely on projection, the relation of the raw data acquired by a CT scanner to the final image isn't completely straightforward. The data acquired through hundreds of x-ray projections at different angles around the patient is called a sinogram. In order to produce the actual tomographic slice it is necessary to do a relatively complex mathematical manipulation of the sinogram in a process called CT reconstruction.

There are several algorithms used for CT reconstruction, ranging from exact methods that produce perfect images but that are so mathematically intensive as to be completely unusable in practice, to iterative methods that are also computationally intensive and time-consuming but are acceptable under certain circumstances, to approximate methods that provide reasonable results in a short amount of time and thus are the ones normally used in practice.

Most of the reconstruction methods typically used are related to the Filtered Backprojection (FPB) algorithm, in which the value of each pixel is determined by accumulating the contributions made by the detector cell(s) that received the rays that traversed the corresponding voxel in each of the projection views. In a simple backprojection the resulting image is extremely blurry, so a convolution filter that reduces the low-frequency components is applied to the sinogram prior to backprojecting, hence the name filtered backprojection.

As is typical with convolution filters, the actual filtering is done in frequency space where the convolution becomes a simple multiplication. Depending on the specific shape of the filter in frequency space certain properties of the image will be enhanced. For example, a filter that favors the high-frequency channels will be adequate for reconstructing an image in which very small objects and other small features should be highlighted, but the corresponding increase in high-frequency noise may make it inadequate for the reconstruction of relatively clean images of larger objects. A fundamental part of this project is the development of a reconstruction algorithm for multi-arm VRX scanners based on FBP which will be described in Chapter 3.

In the realm of the reconstruction of volumetric data from hundreds of two-dimensional projection views, such as those used in CBCT (Section 2.1.2), the algorithm most commonly used is the Feldkamp (FDK) algorithm, which is a generalization to 3-D of the backprojection methods that also takes into account the beam divergence in the transverse direction [21, 71].

2.2 Image Quality Parameters

The goal of all tomographic medical imaging systems is to provide the physician with useful information about the interior of the patient. Some modalities like CT aim to produce an accurate description of all the structures including the different tissues, fluids, foreign objects, etc. For these modalities the quality of the images is paramount.

There are several parameters used to determine and in some cases quantify the quality of the image. Two of the most important ones are spatial resolution and contrast resolution. Other important parameters of the image quality in CT scanners evaluate the levels of noise as this will greatly affect the accurate interpretation of the image. Additionally, due to the way that the tomographic slice is reconstructed

from the different x-ray projections it is essentially unavoidable that some artifacts are going to be produced, altering the final image.

The spatial resolution of an imaging system describes its ability to produce adequate images of very small objects or objects placed very closely together. Contrast resolution of the imaging system on the other hand refers to its ability to distinguish between very similar values of the physical property being visualized in the image, which in the case of CT is x-ray absorptivity. This is important because good contrast resolution will allow the physician to identify and differentiate similar tissues.

2.2.1 Spatial Resolution in the Spatial Domain

The goal of VRX CT scanners is to provide the best possible spatial resolution within the physical limitations of the scanner for patients of very diverse sizes. Because of this, spatial resolution is the most important image quality parameter that needs to be taken into consideration for the purposes of this work. In this section we will briefly describe the most common criteria used to evaluate and quantify the spatial resolution of a CT system.

Although the spatial resolution of an imaging system is more easily understood in the spatial domain, it is frequently evaluated in the frequency domain. The methods use for this evaluation in both domains are equivalent as they both provide a complete description of the resolution properties of the system [58, 20]. The main criteria used to quantify spatial resolution in the spatial domain is the Point Spread Function (PSF), which is defined as the distribution of pixel intensities in the image of a point source of unit intensity [56]. Although in an ideal system this would correspond to a single point in the image plane, in practical systems the intensity is blurred across a larger area. The PSF of a real system is normally asymmetric, but some systems are close to being isotropic, i.e., they exhibit rotational symmetry [56]. In those cases the analysis is simplified as the PSF can be viewed as a 1-D function.

Unfortunately, in practice the direct measurement of the PSF is difficult. In the case of x-ray systems, for example, the minuscule size required for an aperture to approximate a point source leads to the collection of a low number of x-ray photons within a given amount of time, reducing the SNR. Also, it is difficult to obtain a proper alignment to measure the x-ray intensity exactly at the center of the PSF [56].

Another way to quantify spatial resolution in the space domain is through the Line Spread Function (LSF), which is the distribution of pixel intensities in the image of a long, narrow slit of unit intensity [56]. The LSF can be viewed as the 1-D version of the two-dimensional PSF, and in the case of isotropic systems it provides the same information. In more general systems, it would be necessary to evaluate the LSF in all directions to obtain the same information, but in most cases an adequate description of the system can be obtained by evaluating it in two orthogonal directions (for example in the radial and azimuthal directions with respect to the center of the image). A property of the LSF that provides quantifiable information about the

spatial resolution of the system is the FWHM, or Full Width at Half Maximum, i.e., the width of the function at half of the maximum intensity. Sometimes the full width at other fractions of the height of the LSF are also used as measures of the spatial resolution.

Although the LSF reduces many of the difficulties in measuring the PSF, measuring it in practice is still hindered by the low collection of x-ray photons. For this reason, instead of measuring the LSF directly it is sometimes calculated from a related function called the Edge Spread Function (ESF), which is the distribution of pixel intensities in the image of a long, perfectly attenuating edge. The LSF can be mathematically calculated as the derivative of the ESF, making the determination of either function equally useful [42]. A smooth edge is also easier to produce than a narrow slit, which gives the ESF an additional practical advantage over the LSF.

2.2.2 Spatial Resolution in the Frequency Domain

In the frequency domain the measures most frequently used to determine the spatial frequency of an imaging system are the Optical Transfer Function (OTF) and in the Modulation Transfer Function (MTF). Both functions indicate how accurately the system reproduces information at different spatial frequencies.

The OTF is the Fourier transform of the PSF, normalized (the Fourier transform is divided by its value at frequency zero) because the original definition of the PSF assumes a point source of unit intensity [56]. The OTF is in general complex and asymmetric (although for isotropic systems it is real and rotationally symmetric), so it can be divided into its magnitude and phase components. The magnitude of the OTF is the Modulation Transfer Function (MTF), while its phase is called the Phase Transfer Function (PTF). In systems with very asymmetric PSF's the PTF provides information about a phenomenon called "phase distortion" that produces a visible degradation of the image quality [27]. On the other hand, in the case of isotropic (or nearly isotropic) systems the information provided by the PTF is of little value. In those systems, the MTF is usually considered to be a sufficiently good descriptor of the spatial resolution of the system [58, 20].

The Fourier transform of the LSF provides a 1-D version of the OTF and MTF [42]. As with the LSF, these would need to be calculated in all directions in order to obtain a complete description of the spatial frequency of an arbitrary system, but for isotropic systems one direction provides all the information. The MTF is, of course, always real and nonnegative. Furthermore, in 1-D the MTF is an even function because the LSF of physical systems is always real.

When comparing two imaging systems, if one of them always has a higher MTF curve than the other one, it will also have a better spatial resolution [58]. Another related measure of the resolution is the cutoff frequency of the MTF, normally defined as the spatial frequency at which the MTF first reaches zero. Nevertheless, it is frequent to find systems for which the MTF approaches but does not reach zero. In

such cases, a small value larger than zero is used to define the cutoff frequency. The area under the MTF curve between zero and the cutoff frequency is also used as a measure of the spatial resolution of the system.

The spatial resolution in CT scanners is sometimes visually assessed by analyzing images of objects with known geometrical configurations, such as bar patterns of different spatial frequencies. This method is simple and quick and can even be used to produce an estimate of the reconstruction MTF if enough frequencies are represented [20]. Nevertheless it is rarely used in studies where accurate, objective measure of the spatial resolution is fundamental, because it is very difficult to extract accurate data from it, and because the number of frequencies required to obtain a more complete description of the spatial resolution of the system makes it impractical [19]. In practice, the MTF is usually calculated from the LSF, whether the LSF is determined directly or by differentiating the ESF.

2.3 Variable Resolution X-ray (VRX) CT Scanners

In this dissertation we are mainly interested in a specific type of CT called Variable Resolution X-Ray Computed Tomography, or VRX CT. VRX CT scanners attempt to bridge the gap that exists between micro-CT scanners with high resolution but a very small field of view that makes them useful only for imaging small specimens, and clinical scanners that are designed to fit fairly large adult patients but have a resolution that is too low to image a small child or smaller specimens optimally. For this, VRX scanners take advantage of the projective compression principle which states that from the point of view of the x-ray source, the cells in a detector will appear smaller and closer together in projection than their actual size and spacing [17, 14].

2.3.1 X-Ray Detectors in VRX Systems

Two fundamental ways have been envisioned to construct an x-ray detector that could take advantage of the projective compression principle. The first method is straightforward: a continuous or discrete detector can be tilted with respect to the x-ray beam thus improving the resolution by a factor of $(sin(\theta))^{-1}$, where θ is the angle between the detector surface and the x-ray beam. Thus, for smaller values of theta the resolution increases dramatically. Unfortunately, theta can only be reduced to a certain point due to the limited signal intensity, crosstalk between adjacent cells (i.e., x-ray photons that should be detected by a particular cell may traverse it and get detected by the wrong cell), and eventually by a reflection of the x-ray photons on the surface of the detector. This last phenomenon is fortunately an issue only for submicroscopic angles, but the other two problems remain an issue. Several different VRX scanner prototypes have been implemented following the detector angulation method including those employed in this project, and a substantial amount of work

has been done to study the viability of such a detector for CT imaging [14, 16, 17, 18, 40]. In the next sections we will describe several of these implementations.

The second method devised to exploit the projective compression principle is by building a special detector in which the cells are organized in a "stair-stepping" configuration, i.e., in a way that resembles a ladder [14]. The apparent size of the detector cells can be controlled by varying the height the size of the exposed. Although building an x-ray detector with precise miniature moving parts can be a challenge, this method overcomes some of the drawbacks of the "tilting" VRX approach, among them eliminating any possibility of x-ray reflection on the surface of the detector. The implications and implementation of the stair-stepping configuration are the subject of ongoing research.

2.3.2 Single and Dual Arm VRX

All the VRX scanner prototypes that have been built so far employ the tilting method. Some single slice prototypes employ a single linear array of detector cells, or arm, that rotates around a pivotal point (usually at the center of the arm), providing a straightforward application of the projective compression principle.

It has been determined that employing several shorter arms can provide some advantages. The simplest implementation of this is a symmetric dual arm configuration in which the two arms are placed in a V shape, facing the interior. Both arms have a pivot point at the vertex of the V which is located directly in front of the x-ray tube. The symmetry of this configuration helps reduce the inconveniences produced by the fact that in a single-arm system the cells at their "closer" side of the arm appear bigger than those in the "far" side, a fact that becomes more apparent in systems with a wide fan beam, i.e., those in which the distance from the focal spot to the detector is not much larger than the length of the arm. In a symmetrical two-arm configuration, both arms provided the same increase in spatial resolution [14, 15, 51].

2.3.3 Two and Four arm VRX in Targeting Mode

If the arms in the two-arm VRX system are tilted to different angles, the one with the smallest angle will provide a higher resolution, while the one with the largest angle will provide a larger field of view. By combining the information from the two arms, we will find that while we can image relatively large objects (up to the size of the FOV determined by the arm with the larger angle) a circular region around the center of the image will be reconstructed at a higher resolution that can only be achieved in a single arm or a symmetrical dual arm configuration by sacrificing part of the field of view. A system with this property is called a targeting VRX scanner, the interior, higher resolution region is called the target region, and the rest of the FOV is called the external region [13, 15, 51, 23].

The advantages of having both a higher resolution target region and a symmetric system can be obtained by employing four arms instead of two. The arms will be located symmetrically, with the two "inner" arms tilted to a small angle while the two "outer" arms are placed at the ends of the inner arms, at a larger angle relative to the system centerline. The inner arms provide the increased spatial resolution of the target region, while the outer arms provided the large field of view [13, 15, 51, 23].

2.3.4 VRX with Two-Dimensional Detectors

In addition to the single slice VRX scanners described above, a multiple slice VRX cone beam CT system has also been developed [10, 11]. That system employs a CsI-based amorphous silicon flat panel detector similar to those used in some CBCT systems like the ones described in Sections 2.1.2 and 2.1.3, allowing for the acquisition of a large number of slices in a single rotation. The flat-panel detector in the VRX CBCT system can tilt about its horizontal axis and vertical axis independently, increasing the resolution both in the imaging planes and in the perpendicular direction.

Although the VRX CBCT system opens exciting possibilities in the field of VRX imaging, this work is focused primarily on single slice VRX scanners with one, two, and four arms, since we are interested principally in the effects produced in the imaging plane by the application of the projective compression principle.

Chapter 3

Reconstruction and Calibration of Multi-Arm VRX Scanners

The main goal of this project was the development and application of some computerized tools that aid in the design and implementation of Variable Resolution X-Ray (VRX) CT scanners. As described earlier, this document is divided in two parts: the description of the tools developed, and their application to a small series of experiments in which those computational tools proved to be fundamental in the analysis and evaluation of some aspects of VRX imaging. In this chapter we will describe some of those tools, specifically a reconstruction algorithm for multi-arm single-slice VRX scanners, a calibration algorithm for those devices, and other algorithms related to the acquisition and reconstruction of VRX images. In Chapter 4 we will describe the rest of the computational tools, in particular a computer model of multi-arm VRX scanners that has proven to be useful in the study of such devices, and in Chapters 5 and 6 we will present two applications of these tools.

The process of producing an image with any CT scanner includes two overall steps: the physical acquisition of all the projections, and the reconstruction of the tomographic slice from these projections. Figure 3.1 shows more details of the workflow followed in single-slice VRX imaging.

The development of our current multi-arm VRX reconstruction algorithm is described in Section 3.1. In order to reconstruct an image acquired with a VRX scanner, it is necessary to know the *configuration* of the system, i.e., the number of arms and the position of every cell on each arm, as well as its *calibration*, which gives us the exact position and orientation of all the elements in the system (the focal spot of the x-ray tube, the center of rotation, and each detector arm). The calibration also specifies the number of views in a full revolution of the rotary table.[1] While the configuration of the system is straightforward, the calibration is more involved since a small error in the position of any element of the system will affect the quality of the reconstructed image considerably. Single-slice VRX scanners are calibrated using an algorithm described in Section 3.2.

[1] In all prototypes of VRX systems the subject is positioned on a horizontal table that rotates to acquire all the projections. As explained in Section 2.1.3, this rotationg table architecture is commonly used in prototypes of scanners that explore new technologies, as it is far less complicated and expensive than placing the x-ray tube and the detectors on a rotating gantry as is found in clinical and other commercial CT scanners.

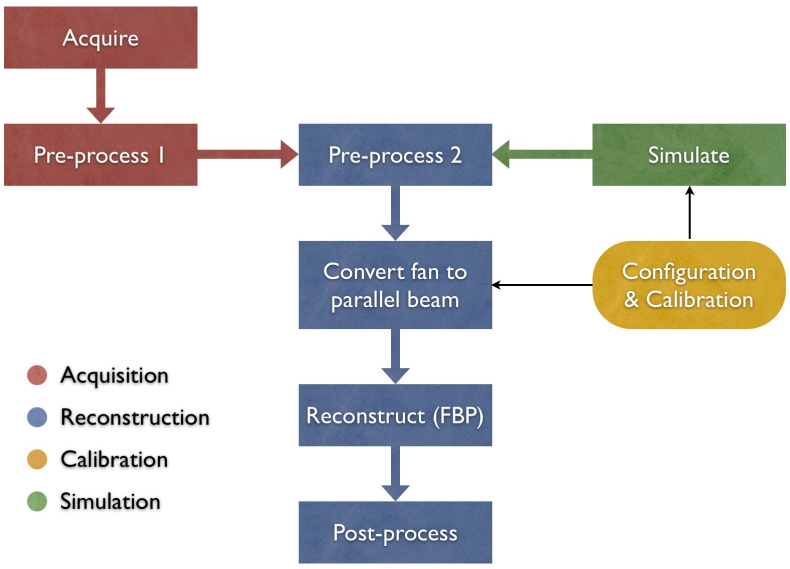


Figure 3.1: **VRX imaging workflow.**

An artificial sinogram can also be produced by a VRX simulator. The description of the geometry of the system is the same in VRX simulators as in the reconstruction algorithm. Therefore, a simulator that can use the configuration and calibration of a real VRX system is an invaluable aid in the study of both existent and possible future VRX configurations. The main VRX simulator developed so far is described in Section 4.1.

3.1 Development of Reconstruction Algorithms

In Section 2.3 we described several different types of single-slice VRX scanners that have been designed and in most cases implemented. The initial versions had single-arm or symmetric dual-arm configurations. In each case, custom reconstruction and calibration algorithms were written to accommodate the specific particularities of each configuration [35, 34]. This unfortunately meant that extensive modification of the code was required after the introduction of new configurations or changes in the parts used (e.g., changing the internal configuration of the linear array detectors used as arms).

When the four-arm VRX configuration was introduced [13], it became apparent that rather than practically rewriting the code to produce yet another inflexible set of algorithms, our efforts should be centered on developing comprehensive and flexible algorithms that would easily accommodate all existing and future VRX models (or at least all single-slice ones) regardless of their configuration and the specifications of the parts used. In the remainder of this Chapter we will describe the main "universal"

VRX algorithms related to the accurate reconstruction of images acquired with single-slice VRX scanners regardless of the number of arms and their specifications.

These algorithms have been used to reconstruct images from sinograms produced both by physical devices and by virtual devices described in Chapter 4. The physical devices have been implemented with different linear detector arrays. One-, two-, and four-arm VRX scanners have been built from four detector segments composed of six modules with 24 $CdWO_4$ detector cells each, for a total of 576 cells. This results in a uniquely irregular cell distribution which must be taken into account for optimal reconstruction. One-arm VRX scanners have also been implemented through the use of an X-Scan f2-307 HE (linear array detector with $CdWO_4$ scintillators, Detection Technology, Inc., Ii, Finland) and individual rows of cells from a PaxScan 2020 (amorphous Si flat panel detector with CsI scintillator, Varian Medical Systems, Salt Lake City, UT, USA).

3.1.1 Overview of the VRX Reconstruction Workflow

Figure 3.1 shows the workflow followed for the acquisition and reconstruction of images with VRX scanners. The physical acquisition is performed by one or more linear x-ray detectors which we call the arms of the VRX system. The signals from all the detectors at each projection angle are concatenated to produce a single VRX sinogram. This sinogram may have some particularities specific to the detector(s) used, like malfunctioning channels, detectors with different sensitivities (a major source of rings in the reconstructed image), etc. Therefore, the sinogram is usually pre-processed either by the proprietary software of the detector or by an external program written specifically for that system before the actual reconstruction phase begins. The raw information obtained after the first preprocessing step is saved in a file with a format that may be specific of the acquisition system (if the preprocessing is done by that system) but the information itself is ready to be processed by the code that is common to all the systems.

The reconstruction process is preceded by any preprocessing of the sinogram that was not done in the previous step. For example, some preprocessing programs will produce a signal proportional to either the photon count or the total photon energy as opposed to the *logarithm* of such value, as required by the reconstruction algorithm, and this must be corrected.

Unlike the older VRX algorithms which reconstructed along a fan beam [35], in the current VRX reconstruction algorithm the image is produced using a simple filtered backprojection method that assumes that the beams are parallel and regularly spaced. This is done because it allows a faster implementation of the actual backprojection, but assuming parallel beams would be an unacceptable approximation of the fan beam found in actuality. Furthermore, the angles between the beams in a VRX system are far from constant. Because of this, an algorithm was developed that derives the *desired* sinogram from the one acquired. Each projection of the result-

ing sinogram is then filtered in frequency space using a Shepp-Logan filter [63], and backprojected to reconstruct the slice.

The reconstructed image is normalized before saving. Additional post-processing steps are applied in some cases, in particular an algorithm for correction of ring artifacts (see Section 3.1.4).

3.1.2 Reconstruction Algorithm for Single-Slice Multi-Arm VRX Scanners

In order to reconstruct images acquired with the different VRX prototypes and configurations, it was necessary to develop a special CT reconstruction algorithm that would take into account the different number and unusual orientation of the arms. Unlike previous VRX algorithms, this one allows any number of arms located in arbitrary positions, and therefore works for all one-, two-, and four-arm single-slice VRX scanners, including asymmetrical two-arm scanners used for target imaging. Furthermore, this algorithm uses a conversion of fan to parallel beams, unlike the previous algorithms which backprojected the fan beams directly [35]. The program code for the reconstruction algorithm developed for this project is listed in Appendix A.

Figure 3.2 shows a typical sinogram acquired with a four-arm VRX scanner. For reasons that will become apparent later, it is usual to acquire projections for more than 360° plus the fan beam angle. The algorithm will eventually use only the central part of the sinogram, which in Figure 3.2 is the region between the green lines. Each pixel of the sinogram represents the signal acquired by one detector cell in one projection view.

Each row of this sinogram can be associated with a specific cell, or alternatively, as shown in Figure 3.3(a) , with the angle β formed by the central line (the line that passes through the focal spot and the center of rotation C_r) and the line passing through the cell and the focal spot F_s. This fan angle β will be negative for cells located to the left of the central line (from the point of view of the tube), and positive for cells located to its right. Similarly, each column of the sinogram can be identified with either the time at which the projection was taken, or with the corresponding angle ϕ of the rotary table.

We can therefore view the sinogram as a function of β and ϕ, and every point in it can be linked to a ray similar to the red line segment in Figure 3.3(a) for the corresponding ϕ. Each of these rays will traverse a portion of the field of view, tracing a chord through it. The conversion of fan to parallel beams is based on the fact that at some time during the rotation of the table, that chord will be parallel to the F_sC_r axis, as shown by the blue line in Figure 3.3(b). This happens when the rotary table is at an angle θ given by:

$$\theta = \phi + \beta. \qquad \text{(Eq. 3.1)}$$

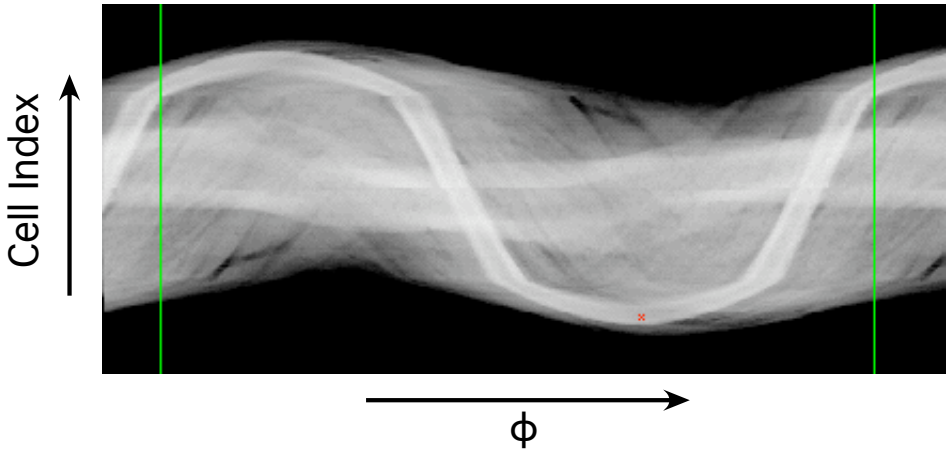
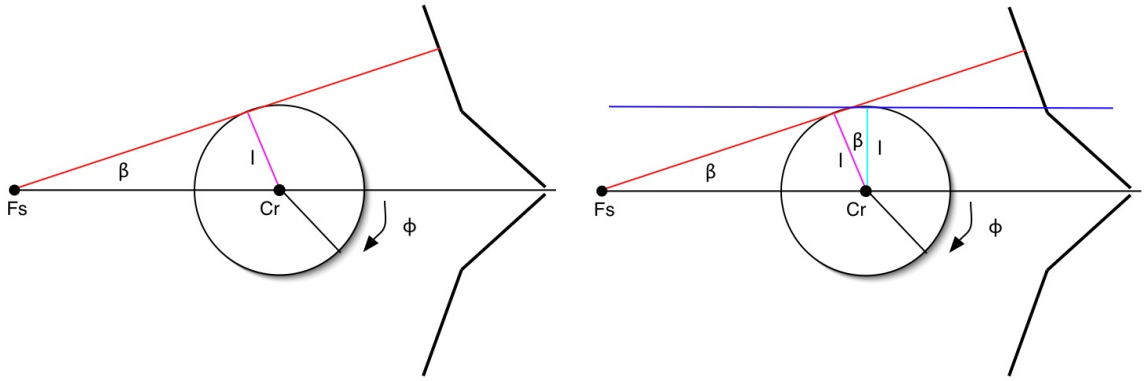


Figure 3.2: **Four-arm VRX sinogram.**



(a) (b)

Figure 3.3: **Schematic representation of a four-arm VRX system**. (Not to scale.) (a) The red line represents a beam from the focal spot F_s to a cell in the leftmost arm, forming an angle β with the F_sC_r axis. (b) If the table is rotated an additional angle β (which may be negative), the beam of figure (a) will become parallel to the F_sC_r axis.

The distance from C_r to the chord is, of course, unchanged, and is given by:

$$l = \overline{F_s C_r} \sin(\beta), \qquad \text{(Eq. 3.2)}$$

where $\overline{F_s C_r}$ denotes the distance between both points.

Equations 3.1 and 3.2 allow us to transform a VRX fan beam sinogram into a parallel beam one. Unfortunately, for normal filtered backprojection we need a sinogram where the parallel beams and the view angles are both regularly spaced. Clearly the sinogram obtained does not meet either condition, as shown in Figure 3.4, a condition that is worsened when the distribution of cells on the arm itself is not regular. For a fixed β (i.e., a particular detector cell), all points (β, ϕ) of the original sinogram will be mapped to the same value of l, thus the distribution in columns observed in the figure. Nevertheless, note that the distance between these columns is far from constant, and that the distribution in the θ axis is even more irregular.

This lack of regularity in the positions of the points of the sinogram is solved by finding an estimate of the value of the signal at each desired combination of θ and l in a regular grid. The algorithms available for this are either designed to solve the opposite problem (find an estimate value for arbitrary points by interpolating on the known values for a regular grid) or extremely computationally intensive (because they assume that the original grid is completely arbitrary). Therefore, a custom algorithm was written that finds each desired value by calculating a bilinear interpolation of its four nearest neighbors from the irregular sinogram. This has been optimized by applying our knowledge of the structure of the irregular grid, along with some clever computational tricks, to the extent that the calculation is done in around one second even for very large sinograms using ordinary hardware. The convenience of acquiring slightly more than 360° becomes apparent here: if the extra views are not available, some of the points at both ends of the regularized sinogram will not have all four nearest neighbors, introducing a small bias in the calculations that in some cases produces almost imperceptible artifacts.[2]

3.1.3 Targeting VRX Scanners

The spatial resolution of an image obtained with a 1-arm or a symmetrical two-arm VRX scanner is not really constant, as the projected width of the cells becomes smaller for cells that are farther away from the focal spot. Nevertheless, in practice the difference in resolution in different regions of the image is not very significant. On the other hand, if the arms of a two-arm VRX system are placed *asymmetrically* around the main ($F_s C_r$) axis, or if a symmetrical four-arm system with the inner arms located at smaller half-opening angles is used, the central region of the image (called the *target zone*) will have a considerably higher spatial resolution, while the system maintains a relatively large field of view [13]. Figures 3.5 and 3.6

[2]These artifacts were originally discovered in "noiseless" simulations. This illustrates a case where the simulator was key in helping improve other algorithms.

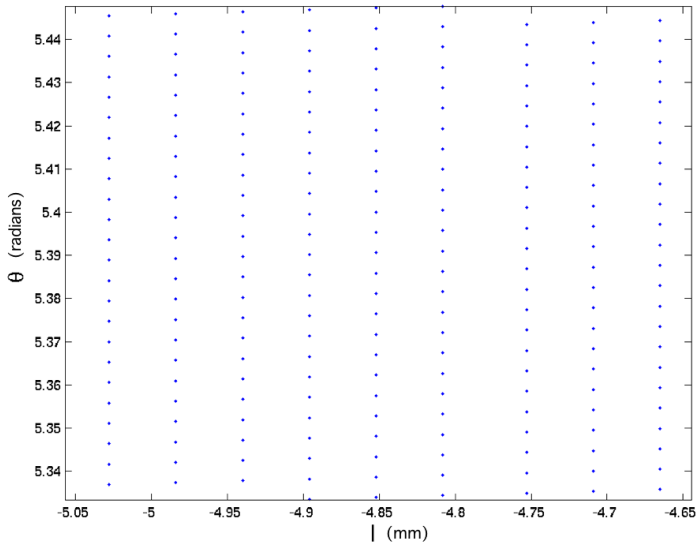


Figure 3.4: **Irregularity of VRX sinograms.**
The plot shows the positions of some of the points in a small section of an (l, θ) sinogram obtained by applying Equations 3.1 and 3.2 to a (β, ϕ) sinogram. Note that even though there is some structure, the points are distributed irregularly in both l and θ.

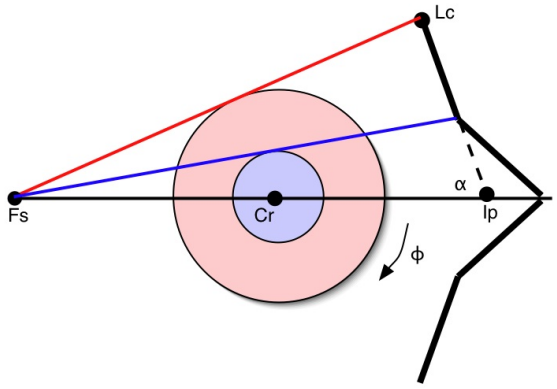
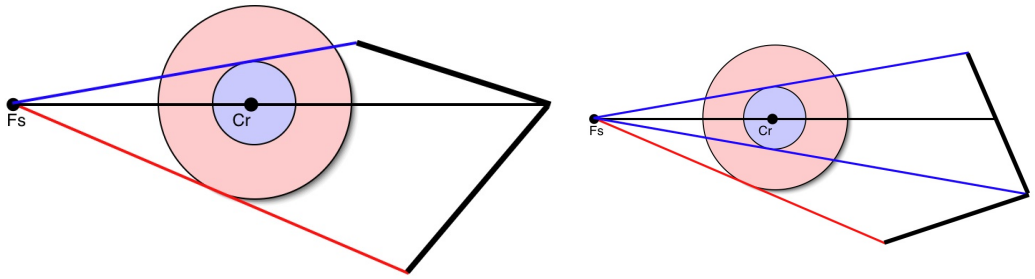


Figure 3.5: **Schematic representation of a four arm VRX CT scanner**.
(Not to scale.)



(a) (b)

Figure 3.6: **Two-arm VRX CT scanners in targeting mode**.
(Not to scale.) (a) Type A. (b) Type B.

show schematic representations of a four-arm targeting system, as well as two different two-arm targeting configurations. The target region can be regarded as the field of view of the high-resolution arm(s), i.e., the arm(s) with smaller half-opening angle(s). Figures 3.7 and 3.8 show the central part of an image acquired with a four-arm VRX scanner.

The procedure described in Section 3.1.2 can be used to reconstruct a sinogram acquired with a four-arm symmetrical scanner. Asymmetrical two-arm scanners, on the other hand, present a problem that requires a modification of the algorithm: points in the external zone (pictured in pink in Figure 3.6) will at some time exit the field of view. Thus, the sinogram will appear truncated, as the example shown in Figure 3.9(a). The image cannot be properly reconstructed with such a sinogram, but since the acquisition is done over at least 360°, the missing information can be deduced from the existing one; that is, for each beam of the missing sinogram there is a corresponding beam acquired with the low resolution arm of the scanner at a different time, when the phantom is rotated close to 180°. This can be seen as constructing the sinogram produced by a "virtual arm" from the corresponding sinogram of the "real" arm located at the other side of the scanner (see Figure 3.9(b)).

The exact angle that the phantom must be rotated for a particular beam actually depends on the fan angle β of the beam, and thus it is only exactly 180° for parallel beams. Therefore, the task is simplified immensely by "stitching" pieces of the sinogram obtained *after* applying the transformation described by Equations 3.1 and 3.2, but *before* doing the interpolation that regularizes the grid.

The sinogram obtained this way can be treated in the same way as a sinogram obtained with only "real" arms. Note, on the other hand, that the advantage of acquiring signals for all beam paths twice during a 360° scan is lost for the external region of the reconstructed image.

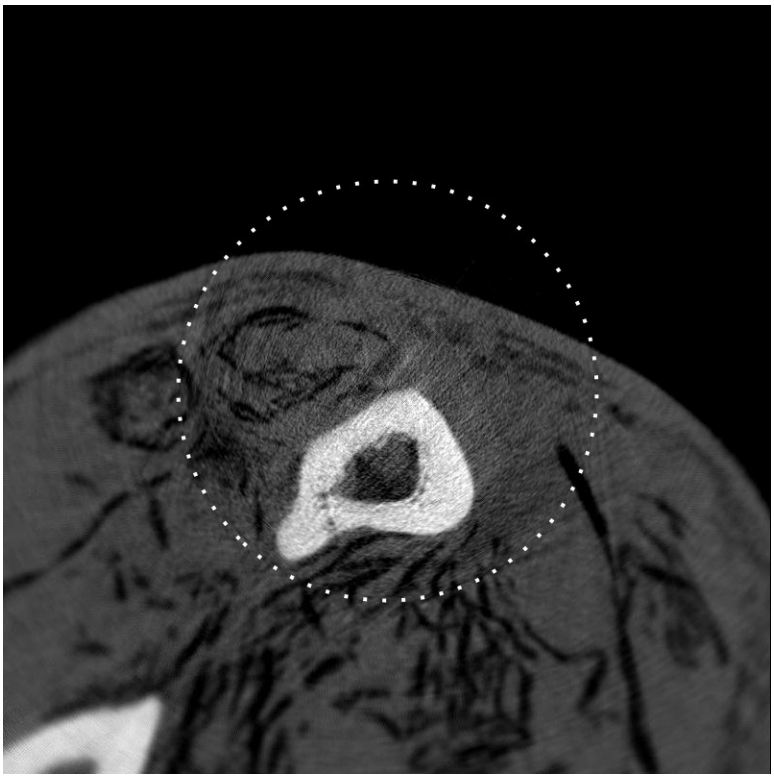
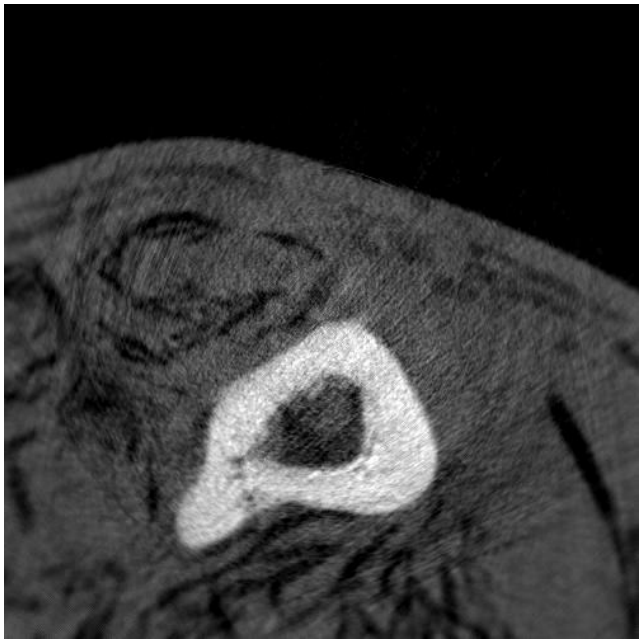
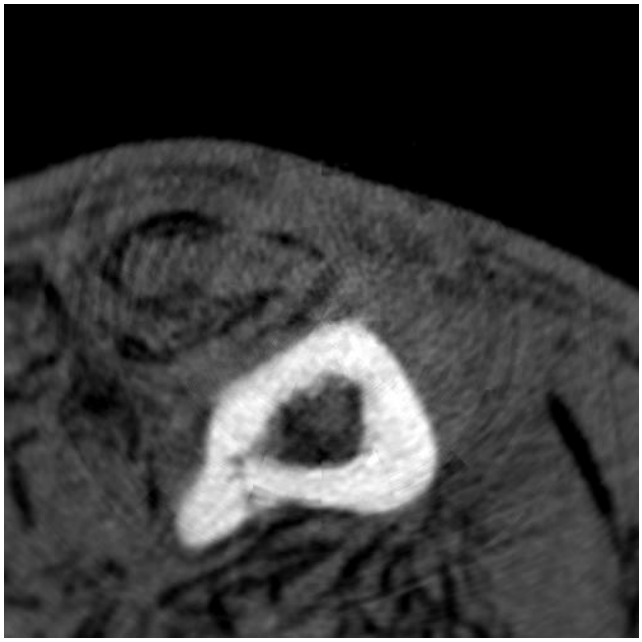


Figure 3.7: **Image of a human forearm cross section obtained with a four-arm VRX CT scanner in targeting mode.**
The target region (highlighted by the dotted line, which is not part of the original image) is imaged at higher resolution, and has a radius of 1.46 cm. The total FOV has a radius of 7.61 cm, but in this image only the innermost 2.6 cm were reconstructed.

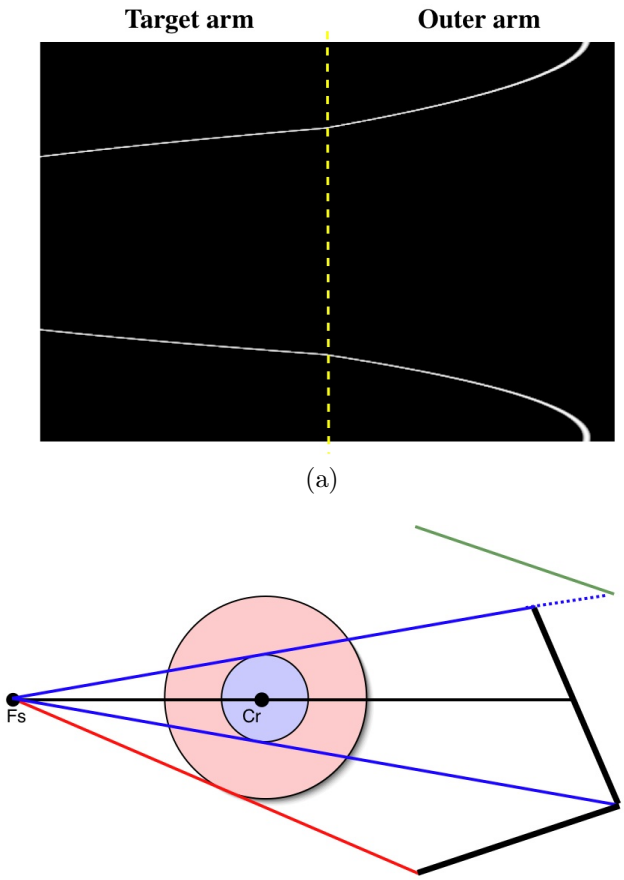


(a)



(b)

Figure 3.8: **Increase in spatial resolution in target imaging**.
(a) Detail of the target region of Figure 3.7. (b) Image obtained by sub-sampling the sinogram to match the arms of a two-arm VRX scanner with similar detectors, therefore showing the resolution expected from such a device. The improvement in resolution in the four-arm configuration is evidenced by comparison.



(a)

(b)

Figure 3.9: **Missing sinogram in asymmetric two-arm VRX.**
(a) A pin located in the external region exits the field of view of both arms during part of the trajectory, as seen in this sinogram (time progresses in the vertical axis). (b) Information from the low resolution arm is used to generate the sinogram from a "virtual arm", pictured in green.

3.1.4 Image Post-Processing

There are several post-processing steps that may be applied to the tomographic image after it is reconstructed. First, the image is normalized to produce pixel values that are either physically significant (like Hounsfield units) or optimized for the file format in which it will be stored. The image can also be modified to enhance features observed at certain pixel value ranges, either by specifying default window level and width or by modifying the image brightness and contrast.

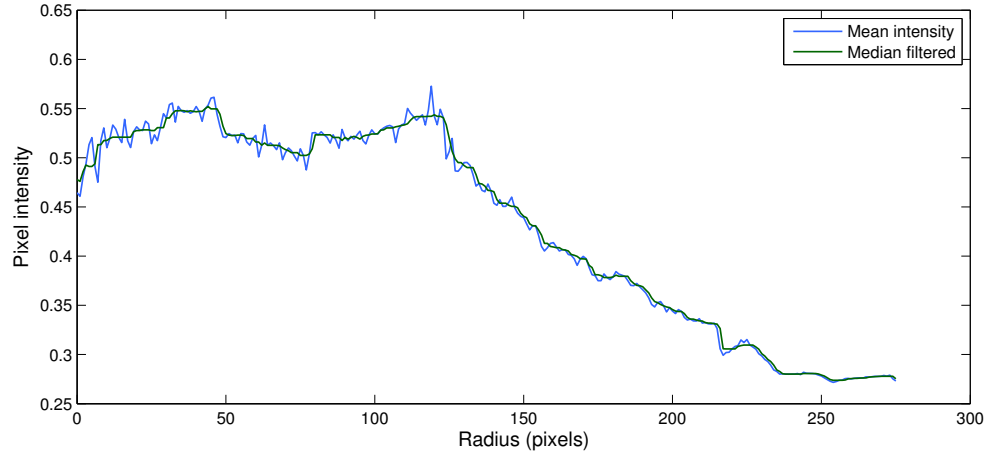
Additionally, some detector technologies are particularly susceptible to generating cell signals with different biases, so some ring artifacts may be produced even after normalizing the sinogram in the pre-processing steps. These residual ring artifacts can be significantly reduced in post-processing. For this, the image is divided into thin concentric rings and all the pixels in each ring are multiplied by specific factors that will attenuate the ring artifacts. These factors are found by calculating the mean pixel value for each ring and interpreting those mean values as a function of the radius of the ring (see Figure 3.10). Ring artifacts in the original image are reflected as spikes in this curve: positive spikes correspond to bright rings and negative spikes to dark ones. The desired effect of the factors is to alter the mean pixel values of each ring so the resulting curve is smoothed. A "desired" or "target" curve can be found by applying a smoothing filter to the mean curve. In Figure 3.10, the original mean curve (in blue) is smoothed by applying a median filter to it. The result is shown in green. The ratio of these two curves will give us the desired factor for each value of the radius.

The ring correction algorithm described here was further improved by dividing each ring into several arcs, and calculating mean pixel curves for each of the resulting wedges independently. Sample reconstructed images before and after such ring correction are shown in Figure 3.11, and an example of the processing of one of the wedges is shown in Figure 3.12.

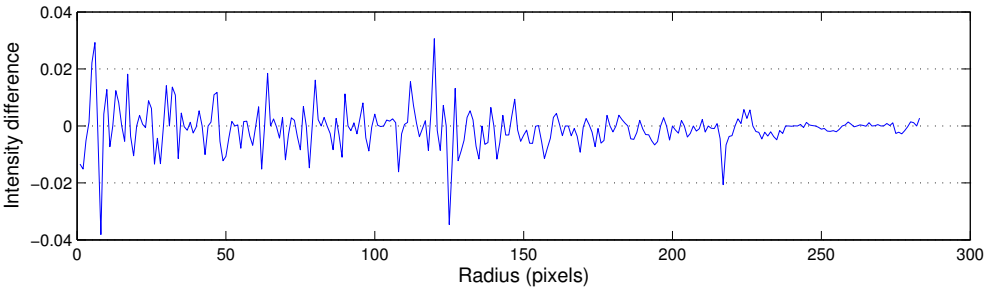
3.2 Calibration of Single-Slice VRX CT Scanners

The quality of an image acquired with any CT scanner will be affected considerably by miscalibration of the system. In multi-arm VRX scanners it is particularly important to know precisely the position of each arm and where it is located relative to the focal spot and to the center of rotation, as a small error in any of these values will cause a significant mismatch between the backprojections of that arm and those of the other arms.

Calibration in single-slice VRX scanners implies determining, for each arm, the half opening angle (α in Figure 3.13), the distance from the center of rotation to the intersection of the main axis and the arm line ($\overline{C_r I_p}$), and the distance of the leftmost cell to that intersection point ($\overline{L_c I_p}$), in addition to the distance from the focal spot

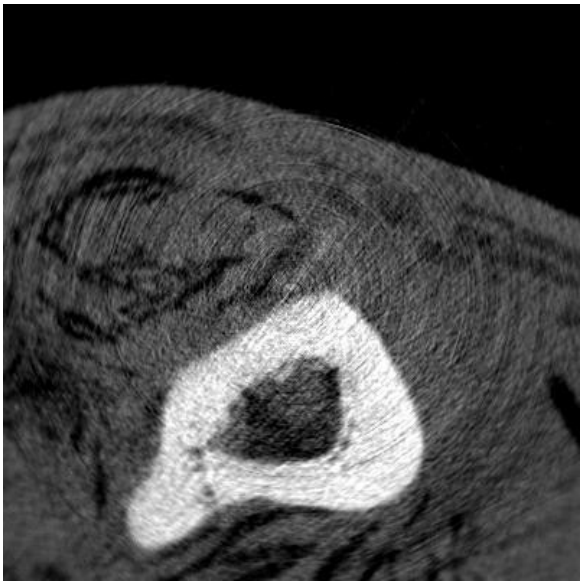


(a)



(b)

Figure 3.10: **Correction of rings in post-processing.**
(a) The blue curve shows the mean pixel value for concentric rings in a reconstructed image. Ring artifacts in the image correspond to spikes in this curve. The curve is smoothed by applying a moving median filter to it, as shown in green. (b) The difference between the two curves shows the location and severity of the rings, with positive spikes corresponding to bright rings and negative spikes to dark ones.



(a)

(b)

Figure 3.11: **Example of ring correction.**
(a) Central region of the image shown in Figure 3.7, before ring correction. (b) Ring correction was applied using concentric one-pixel-wide rings each divided into 16 arcs.

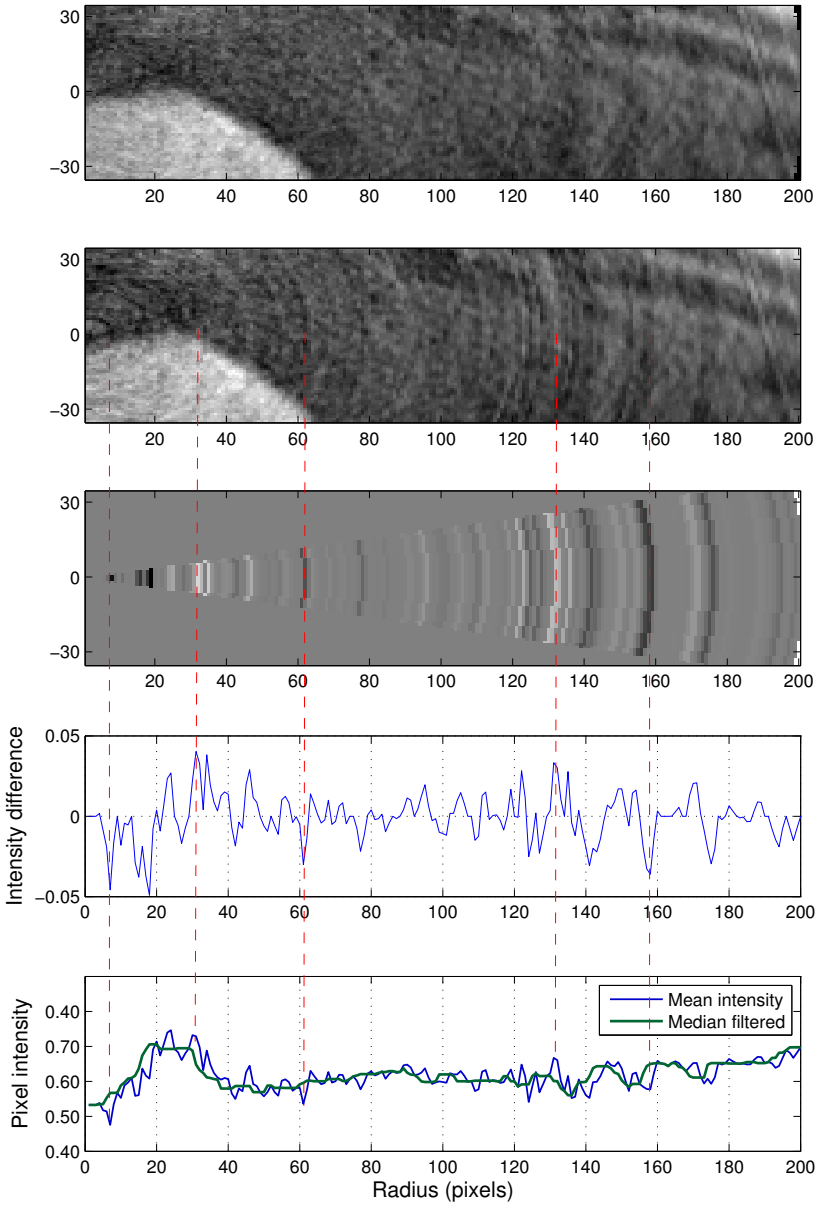


Figure 3.12: **Example of ring correction on an image wedge.**
The top two images show a region of Figure 3.11(a) after and before ring correction. The correction on a wedge–shaped ROI was achieved by subtracting from the original image the arcs shown in the third image. The intensities of these arcs, in turn, correspond to the difference (pictured in the fourth plot) between the curve of mean intensities of the members of each arc and the median filtered version of that curve (both shown in the last plot). The red dotted lines show the positions of the effects of some representative arcs in the different plots.

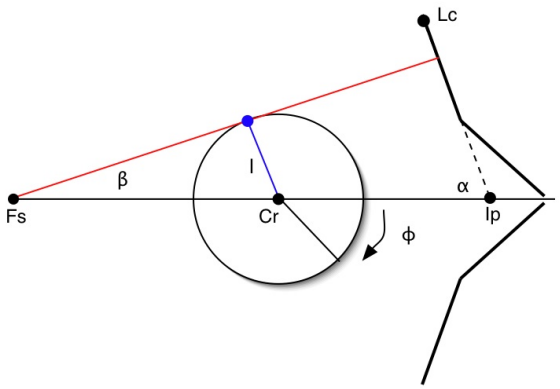


Figure 3.13: **Schematic representation of the calibration setup.**

to the center of rotation ($\overline{F_s C_r}$). Thus, for calibration of an n-arm VRX system it is necessary to determine $3n + 1$ parameters.

The calibration procedure currently used for VRX scanners is based on a procedure originally described by Jordan et al. [34], adapted to accept an arbitrary number of arms. The algorithm follows an iterative process in which an initial estimate of the $3n + 1$ parameters that need to be determined is expected to converge towards their correct physical values.

For this, a sinogram of a thin metallic pin placed at a precise known distance from the center of rotation is acquired on the physical VRX scanner. In a traditional CT scanner such a sinogram will look like a sinusoidal curve (hence the name), but in VRX scanners, and in particular multi-arm models, that curve will appear significantly distorted. An example of such a curve is pictured in blue in Figure 3.14. For each view (projection) number, we can calculate precisely where the shadow of the center of the pin is expected to be located on the detector arm(s) according to the initial estimate of the calibration parameters.

If the parameters are correct, the corresponding expected curve will match perfectly with the real one. In reality both curves are normally very close, but rarely coincide because the initial estimates of the parameters are derived from measurements that are inherently imprecise. An additional source of mismatch between the curves is due to a possible error in the phase, i.e., the angular position of the pin corresponding to each view.

Figure 3.14 shows the curve for the initial guess in red. In that particular case one of the arms (an interior arm, the second one from top to bottom) appears to be correctly calibrated, while the other three arms are not, in particular the exterior arm corresponding to the lowest part of the curves. It is important to note that the system is extremely sensitive to small errors in the calibration (in particular for arms with very small VRX angles) so a visual comparison of the two curves is far from sufficient. In fact, what may appear to be a small miscalibration of the inner arms

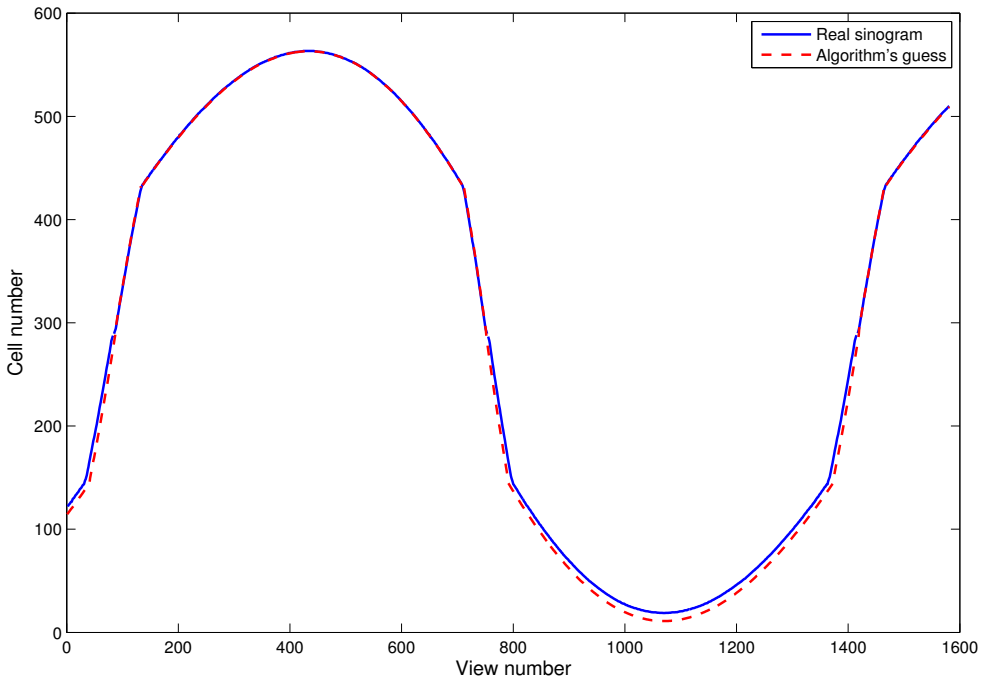


Figure 3.14: **Real vs. expected pin sinogram in VRX calibration.**
The blue curve shows the positions of the center of the shadow of a rotating pin imaged with a four-arm VRX scanner. The red curve shows the expected positions according to the current calibration parameters of the system. The calibration algorithm adjusts those parameters until the red curve matches the blue one.

in Figure 3.14 actually corresponds to errors in the initial guess of the position of the pin's shadow of up to 20 cells.

The purpose of the calibration algorithm is to obtain values for the calibration parameters that are closer to the physical ones by trying to minimize the error between the expected (red) curve and the real (blue) one. For this, a measure of the error is needed, and the current version of the algorithm uses the square root of the average of the squares of the individual errors for each view. This error value will therefore be a fairly complex function of $3n + 2$ variables: the $3n + 1$ physical parameters of the n-arm VRX scanner that we want to optimize and the phase angle mentioned above. It should also be zero for a perfectly calibrated system, so we must find the combination of those parameters for which that function reaches its minimum. Figure 3.15 shows how the error function decreases with successive iterations of the algorithm. A reduction of the error of almost three orders of magnitude as shown in this example is typical. In the current version of the calibration algorithm this is done by an implementation of the Nelder-Mead (or Simplex) algorithm [45].

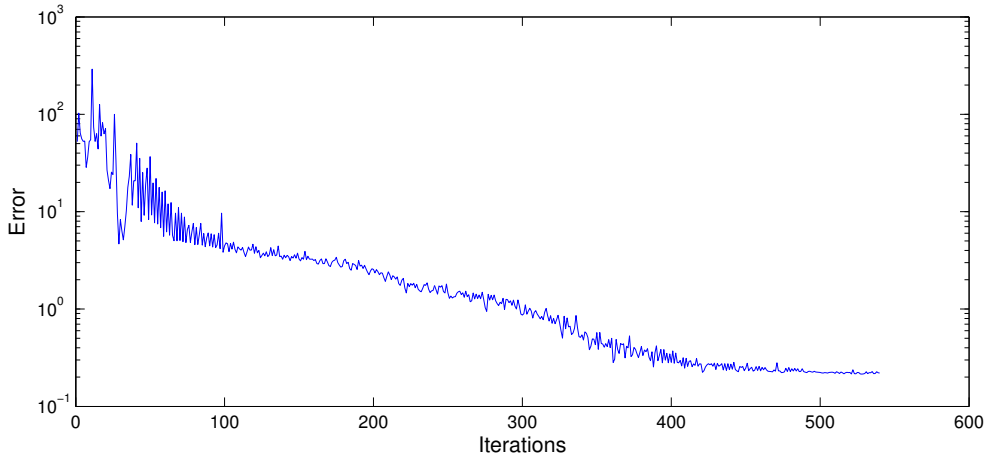


Figure 3.15: **Decrease in the calibration error as the algorithm progresses.** Successive iterations of the minimization algorithm show a decrease in the calibration error of almost three orders of magnitude.

Chapter 4

Computer Simulations of VRX Scanners

The optimization of x-ray imaging devices is crucial as it is important to provide the radiologist with the best diagnostic image possible while minimizing the amount of radiation to which the patient is exposed. All commercial devices are thoroughly tested, but in the development of prototypes, performing physical tests is not always possible or desirable. This has led to the development of computer models of the devices, which have become invaluable for research.

A computer simulator of an imaging device allows researchers to study configurations that cannot be built physically or that would be too costly or time-consuming to build. This happens frequently and for numerous reasons. On one hand, it is frequently important to study numerous variations of a configuration, which would invariably consume a large amount of time. Also, building each configuration may be extremely difficult, and some configurations may require materials and equipment that are not readily available. All this would amount to a substantial increase in the costs, which can rapidly become prohibitive for any lab. A flexible computer model allows all these tests to be performed easily and with minimal additional cost. Computer models also facilitate system development and research, since all the parameters that determine the system behaviour can be isolated. In this way, it is possible to study their effects individually.

The most accurate algorithms available for the modeling of x-ray devices are Monte Carlo simulations such as GEANT, by far the most versatile and sophisticated Monte Carlo program, written at CERN and used by the high-energy physics community [2, 3]. Other packages frequently used for theses purposes include ITS, EGS and PENELOPE [29, 48, 62, 5, 4]. All these programs take into account every event in the system, including the generation of each photon, their interaction with objects in their paths, and the production of secondary particles (scattered photons, light photons, electrons, etc). This is the most accurate way to account for phenomena such as photon scattering. Unfortunately, while Monte Carlo simulations have a well deserved position in many aspects of medical imaging, they are still too computationally demanding to be practical in the production of simulated CT images, as will be discussed in Section 4.1.3.

There are some non-Monte Carlo algorithms publicly available that can be used as practical simulators of CT scanners. Most of them are unfortunately too closely linked to specific designs; for example, they assume particular configurations of the detectors, and those design decisions are deeply embedded in the algorithms. Since

VRX scanners have particularly unique geometries, adapting such algorithms (that is, if the code itself is available) would require considerable effort.

These considerations led to the development of in-house VRX simulators that focus specifically in the physical phenomena that are of most interest to us, that are a perfect fit for simulation of experiments involving VRX-related phenomena, and that are adequate for our computational resources. These simulators are described in Section 4.1, and are a core tool for this project. The code for the main simulator developed, and which was used to generate all the simulations included in this document is listed in Appendix B.

4.1 Development of a Computer Model for Single-Slice Multi-Arm VRX Scanners

Previously we have presented some examples of the usefulness and great value of simulators in the study and development of medical imaging devices in general, and VRX scanners in particular. We have also explained why the previously available packages fail to fulfill our requirements as VRX simulators, in some cases due to an extremely high computational demand, and in others due to a lack of flexibility to allow modeling the particularities of VRX scanners.

These conditions led to the development of our own VRX computer simulators that include all the distinctive aspects of VRX scanners, while modeling only the physical phenomena that we consider most significant, relevant to our research, and that can be handled with our computational resources. The main VRX simulator developed so far is a single-slice multi-arm VRX scanner that can model any VRX scanner that can be described with the same parameters used for calibration and reconstruction detailed in Sections 3.1 and 3.2.

A simulation of a VRX experiment can be divided in three domains: the description of the experimental setup, the definition of the phantom, and the modeling of the physical phenomena considered.

4.1.1 Description of the Experimental Setup

In order to simulate a particular VRX scanner, a model of the specific devices that are used must be defined. This includes describing the position of each element in the system (x-ray tube, center of rotation, and the position and orientation of each detector arm and thus of each cell), and is specified in the same way as a calibration of the corresponding physical device (Section 3.2).

The experimental setup also includes the specification of the x-ray beam in terms of tube voltage and current, and energy distribution of the photons after the intrinsic and any additional filtration (filtration of the beam is not included in the

simulation, so the energy distribution must already include these effects). The energy distribution of the filtered beam, along with the attenuation coefficients of all the materials involved are determined by interacting with the SPECT (spectrum) program.

4.1.2 Definition of a Mathematical Phantom

One of the main challenges of CT scanner simulations is acquiring in a reasonable time a large number of projections, in order to reconstruct an image with reasonable quality.[1] Furthermore, in each view the phantom is rotated to a different angle so all projections must be recalculated. It is therefore desirable to restrict the phantom to objects that can be described by simple mathematical formulas that allow very fast calculations of the intersections of these surfaces and the photon paths. These mathematical formulas must also be suitable for easy rotation to arbitrary angles. For these reasons, most simulators of x-ray devices (not only CT scanners) restrict the phantom to a combination of objects that can be described by line segments and conic curves (for single slice phantoms) or by quadric surfaces and surfaces defined by triangular tessellation (for 3D phantoms).[2]

The VRX simulators developed so far model single-slice devices. The objects that compose the phantom can be of two kinds: *convex polygons* and *ellipses*. A convex polygon is one that has no interior angles greater than 180°, and therefore the line segment connecting any two points in the interior of the polygon will be completely contained within it. In the VRX simulators, convex polygon is defined by the coordinates of its vertices when the rotary table is at 0°. Ellipses, on the other hand, are defined by the position of the ellipse center at 0°, along with the major and minor radii and the tilt of these axes at 0°. The interior of each object is defined to be composed uniformly of a particular material, with attenuation coefficients obtained from SPECT.[3]

If two objects overlap partially or totally the intersection is assumed to be composed by only one of the materials, corresponding to the last object specified in the phantom description file. This implies that if both objects are made of the same material, they can be regarded as a single, more complex object. In this way, complex objects such as concave polygons (i.e., a polygon that contains pairs of points that cannot be connected by a line segment completely contained within it, or alternatively

[1] In traditional CT scanners, a good rule of thumb is to acquire a number of projections that is twice the number of samples in each projection (typically this is the number of cells in the detector array) or more. In practice this means that the number of projections needed for a good reconstruction is at least in the upper hundreds.

[2] *Quadric surfaces* are surfaces described by quadratic equations (and are therefore the 3D equivalent of conic curves). A *triangular tessellation* of a surface is an approximation of it by triangular tiles that don't overlap or leave gaps.

[3] Any material can be modeled by SPECT by specifying its density and the ratio of atoms of each element in the composition.

a polygon that has at least one interior angle of more than 180°) can be defined, as well as objects with both straight and curved edges.

4.1.3 X-ray Production, Absorption, and Detection

The core of the VRX CT simulator deals with the production and detection of the primary x-ray photons. It is important to note that at this time the model does not include photon scattering, a very important source of noise. This is due to technical reasons: the only accurate way to model photon scattering is by doing a Monte Carlo simulation, where each photon is tracked individually, every primary interaction is recorded and all secondary particles that are produced by these interactions are also tracked. In order to perform a Monte Carlo simulation of a CT scan, this procedure must be repeated for all the photons of a large number of views.[4] Very detailed Monte Carlo simulations using complex phantoms may need more than 24 hours to complete on a large cluster of computers totalling 200 modern CPUs, *for a single projection* [5, 4]. For our experiments the phantom does not need to be so complex, and the number of photons for a single projection is not as large, but the gains in speed due to these factors are counterbalanced by the need to simulate hundreds of individual projections. An accurate modeling of the scattering effects in a reasonable time is therefore something that escapes the computational resources available for this project.

An additional, less pressing complication inherent to modeling scattering through Monte Carlo simulations is that the problem must necessarily be solved in 3D space, adding to the complexity of the algorithm. The study of VRX systems based on flat panel detectors will demand the development of simplified simulations in three dimensions, including 3D phantoms. For single slice systems, on the other hand, we prefer to employ a 2D representation corresponding to a flat collimated fan beam, which allows us to employ more precise models for most of the other physical events.

The VRX simulators developed so far do take into account most of the x-ray related phenomena (other than photon scattering):

As stated before, the energy distribution of the beam is determined using SPECT. This means that all the photons considered in the simulation are divided into many energy bins (typically one or two per keV), following the distribution given by SPECT. Both the intrinsic and any additional filtration of the beam are included in the calculation of the distribution.

The beam is supposed to be collimated, producing a flat fan beam. Only the photons for which the initial trajectory falls within the field of view of the scanner are considered. The field of view is determined by the positions of the leftmost and

[4] It is possible to estimate the effect of scattering on some of the views by interpolating the effects found on adjacent views. Nevertheless, this approximation will only be valid if the effects are calculated for a few hundred views [47].

rightmost detector cells. The fan beam is divided into many narrow fan beams, each one being projected to a small portion of a single detector cell. The central ray of each narrow beam is used to represent the path of the whole beam.

For each view, the mathematical formulas representing all the objects in the phantom in the corresponding position are calculated. With these formulas it is easy and computationally efficient to find the intersections of each ray with each of the objects, and the order in which these interfaces are traversed. In this way, each ray is divided into ordered segments, each one corresponding to a single material.

The attenuation of each ray is calculated as it traverses each segment. This is done separately for each energy bin. The attenuation coefficients for each material at each energy level are provided by SPECT.

After the rays exit the phantom and reach the detector, the signal acquired by each cell is determined. In a simple model, this can be done by adding the surviving photons of all the energies for all the beams that reach that cell (if the detector is a photon-counter), or by finding the total energy deposited (if it is an energy integrator). A better model for the detector will take into account cell-to-cell cross-talk by finding the absorption of photons in each of the layers of each detector cell they traverse, in a process very similar to the x-ray absorption in the phantom.

The signals acquired by all the virtual cells in each view constitute a sinogram comparable to those obtained by the real device. Such sinogram can therefore be reconstructed by the algorithms described in Chapter 3 without modification.

4.1.4 Approximation of Scattering Effects

As explained in Section 4.1.3, the most accurate way of accounting for the effects of photon scattering is through Monte Carlo simulations, but that approach is not viable given our constraints. In our current model, the effects of scattering are replaced by a small increment in the Gaussian noise that is added to the sinogram, an approach that is far from accurate. This section describes a method that we believe will be a much better approximation of the real phenomenon.

The method is based on the fact that the number of scattered photons is larger in regions where more interactions occur. These regions appear in the initial sinogram as regions with higher attenuation, where the detected signal is lower. This is the basic algorithm that would be followed to estimate the signal detected from the scattered photons:

- For each projection, invert the original signal detected, so the resulting signal is higher where more interactions occurred.
- Convolve the resulting signal with a wide kernel to obtain a blurred version of the signal.

- Multiply the blurred version by a scaling factor, so that it reflects the intensity level that would be expected of the scattered photons.
- Add the resulting signal to the original projection.

This method requires some calibration in order to obtain the scaling factor that best mimicks the correct physical effects of scattering. This calibration would be done earlier by comparing the blurred signal for some representative cases to the signal produced by scattered photons obtained with a Monte Carlo simulation. Note that the scaling "factor" is unlikely to be constant.

4.1.5 Other Simulation Details

Other physical phenomena covered by the simulation include the addition of noise to the acquired signals. This accounts for different sources of noise, most notably quantum noise, and approximates the effects of photon scattering. The amount of noise affecting a particular cell depends on the number of photons reaching it. Thus, the algorithm adds to each signal Gaussian noise with a standard deviation that depends on the overall dose, and the distance from the cell to the tube.

In some of the systems being modeled the focal spot is relatively large. Since this will have a considerable effect on the spatial resolution of the system, it is included in the simulator by doing a convolution-like operation of the projected shape of the focal spot (which changes size at different parts of the detector) and the acquired signal.

It is interesting to note that, even though the simulators and reconstruction algorithms developed for this project were conceived with the particular characteristics of VRX scanners in mind, most of the techniques developed for them are applicable to most CT scanners and, to some extent, to other x-ray imaging devices. As such, the algorithms described here provide unique and fresh ideas that have a reach that goes well beyond VRX imaging.

4.2 Corroboration of the Computational Models

Some of the virtual experiments described in the previous sections cannot be replicated physically in our laboratory without a significant investment of resources that are not available for this project. Others would be extremely time demanding, and therefore not all of them can be performed in the time available. For this reasons, only a subset of the experiments were performed with physical scanners.

The physical experiments included in the project are:

- Basic images of objects of known composition. These experiments allow us to verify the integrity of the acquisition process and the accuracy of the recon-

struction algorithm. Once the images obtained are deemed adequate, they were compared with images of simulated phantoms that replicate the real objects, allowing us to verify the accuracy of the simulation algorithms.

- Images of edge phantoms acquired with a four-arm scanner in the three configurations described in Section 5.2.1, used to verify the spatial frequency results.
- Images of streak artifacts produced by thin metallic foils encased in acrylic (such phantoms have been used in the direct determination of the LSF), and images of beam hardening-related artifacts such as cupping. The images of artifacts were produced with a single-arm device employing an X-Scan linear x-ray detector. Due to time constraints, the images were repeated for fewer VRX angles than those used in the virtual experiments, as they were used mainly to verify the validity of the simulations.

4.3 Optimal Parameters for Simulations[5]

Computer simulators are limited in that they are based on the interaction of imperfect mathematical models of the different parts of the system. In the case of most CT simulators, one of such models is the representation of a true polychromatic x-ray beam as a collection of several monochromatic beams with initial numbers of photons chosen to reflect the initial spectrum of the real beam. Another example is the use of a finite number of rays emanating from the x-ray tube and hitting each detector element at specified points. In these particular cases, a much better approximation of the physical phenomena can be reached by using a Monte Carlo model in which each individual photon is created with a randomly-selected energy and initial trajectory [9]. Unfortunately, the computational intensity of such a simulator makes it impractical for most purposes using current desktop hardware. Furthermore, defects in the mathematical models for other parts of the system will in any case affect these kind of simulators.

Because the original goal of the simulator discussed here was to aid in the design and improvement of VRX scanners and the computational algorithms associated with them, it was designed following a fast, non-Monte Carlo approach, and thus suffers from the imperfections mentioned above. This Section examines the effect produced by both parameters (the number of bins used to represent the polychromatic beam and the number of beams per detector cell) on the quality of the resulting image. This study also examines the impact of a third parameter, the number of view angles in the sinogram. Even though the number of views is an inherent parameter of the real system and not exclusive to the simulator, it is interesting to compare its effect with those of the other two parameters.

[5]Portions of this Section adapted with permission. D. A. Rendon, F. A. DiBianca, and G. S. Keyes, "Computational Models of Variable Resolution (VRX) CT Scanners," *Proc. of Biomedical Science & Engineering Conference*, Oak Ridge, TN (2009).

4.3.1 Computational Models of VRX Scanners

As described in the previous sections, the VRX simulator used here takes into consideration the geometry and characteristics of one or more detectors (arms) and of the x-ray tube. The phantom is composed of any number of ellipses and convex polygons of arbitrary sizes, positions and materials.

The signals acquired by the detector cells in every view of the sinogram are found by calculating the attenuation of polychromatic x-rays as they traverse the different materials on their way to the detector cells. The initial energy distribution of the beam and the attenuation coefficients of the materials involved are obtained from the "SPECT" program (a polychromatic x-ray absorption routine). The output of each simulation is a sinogram with the same characteristics as those acquired with physical devices, so it can be reconstructed using the same algorithms.

In order to simplify the analysis, we modeled a single-arm VRX scanner. In single-slice, single-arm VRX scanners a single linear x-ray detector is tilted around an axis perpendicular to the imaging plane that passes through the center of the face of the detector. Due to the projective compression principle, (i.e., the projected size of an x-ray detector cell will decrease as the detector is tilted), an image reconstructed from a sinogram acquired with the tilted arm will have a higher spatial resolution and also higher quantum detection efficiency since the pathlength in the detection medium becomes longer, at the expense of having a smaller FOV [17, 14, 16].

4.3.2 Experimental Setup

The VRX system modeled for this project uses an X-Scan f2-307 HE, (Detection Technology, Inc., Ii, Finland), a linear array detector with $CdWO_4$ scintillators, 768 individual cells, and a pixel pitch of 400 μm. It also employs an UltraBright Microfocus X-Ray Source (Oxford Instruments X-Ray Technology, Inc., Scotts Valley, CA, USA). In the virtual setup, the tube is set to operate at 80 kVp, 1 mA, 3 mm Al intrinsic filtration.

Figure 4.1 shows a schematic representation of the experimental setup. All the experiments were run at three representative VRX angles: 10°, 20°, and 60°. This produces fields of view with diameters measuring 3.9, 7.5 and 16.7 cm respectively, but we only reconstructed the central squares with sides of 2.6, 5.0, and 11.2 cm. In all cases the reconstructions were made on 1200×1200 pixel matrices.

Three blocks of acrylic of different sizes, one for each VRX angle (measuring 0.5×1, 1×2, and 2×4 cm), were used as Edge Spread Function (ESF) phantoms. One of the long sides of each block represents a surface that ideally would be perfectly polished to flatness and perpendicular to the imaging plane in such a way that it can be used to represent a smooth edge transitioning from air to acrylic. The blocks were located at 0.5, 1, and 2 cm from the center of rotation, with the edge side facing the COR. The phantom was tilted 8° to avoid having the edge parallel to the rows

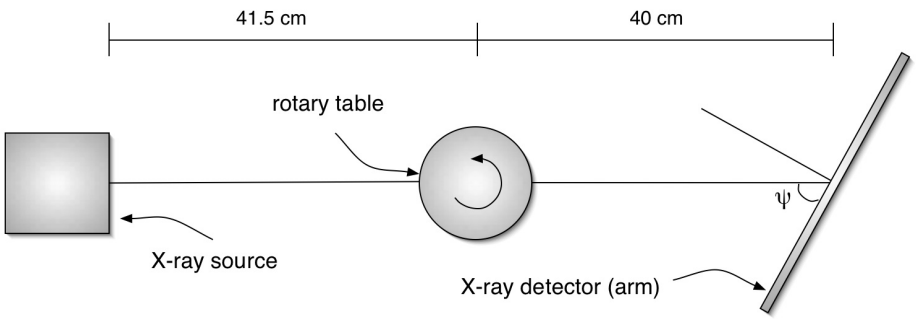


Figure 4.1: **Schematic representation of the simulated VRX CT setup.** ψ is the VRX angle.

or columns of the reconstructed matrix. Figure 4.2 shows an example of a typical simulated image.

4.3.3 Spatial Resolution

One of the long edges of the acrylic phantom was used to approximate a perfect edge in the image. A rectangular ROI including a substantial part of the edge was extracted from the image, and used to calculate an equation for the line that contains the edge. The distances from each pixel of the ROI to that line were then calculated, producing a very densely sampled step function. A sigmoid function was fit to the points, producing a smooth approximation of the ESF, which was then differentiated to obtain an estimate of the LSF. The LSF was in turn used to calculate the normalized MTF. In addition to the MTF curve itself, the spatial frequency at which the curve first crossed the 5% threshold was used as a figure of merit for the spatial resolution of the image produced with that specific configuration.

4.3.4 Comparison with Reference Images

In addition to calculating the MTF of the system, each of the experimental images was compared to a reference image produced with unusually strict parameters. For these images, the energy spectrum was divided into 160 bins, 20 rays were projected to each cell, and 2000 views were taken per full rotation. These values are equal or larger than the corresponding ones for the rest of the images (see Section 4.3.5), and are therefore expected to be closer to an ideal real image. One such reference image was created for each VRX angle. All the other images created for that angle were then compared to the corresponding reference image by calculating the root mean square of the errors for all the pixels:

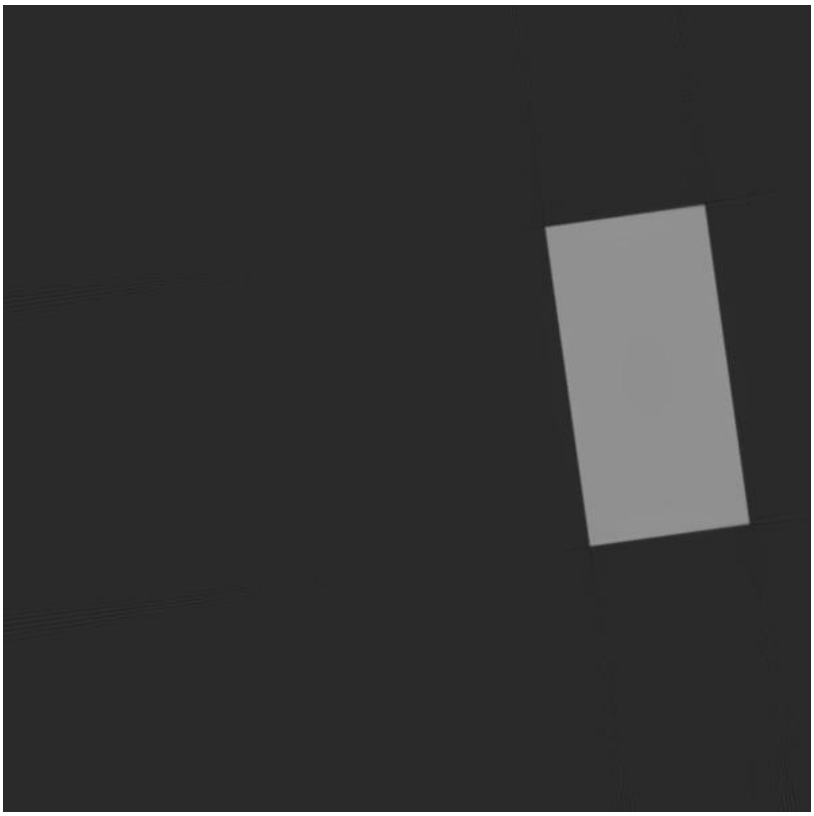


Figure 4.2: **Reconstructed image of an acrylic phantom produced by the virtual scanner.**
The image was generated assuming a VRX angle of 20°, 40 energy bins, 10 rays per detector cell, and 800 views covering a full rotation. The block measures 1 cm $\times$2 cm and the side of the 1200 $\times$ 1200 pixel image measures 5.0 cm.

$$E_{\text{total}} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{j,\text{image}} - x_{j,\text{reference}})^2},$$

where $x_{j,reference}$ and $x_{j,image}$ are pixel values obtained from the reference image and the simulated image being analyzed, and n is the number of pixels in the image.

4.3.5 Virtual Experiments Performed

A large number of simulations for different combinations of the three parameters (number of energy bins, rays per cell, and rotation views) were performed. These were grouped in three main experiments in which two of the variables are kept fixed at typical values, and the third one is varied. The three groups of virtual experiments were repeated for each of the VRX angles studied (10°, 20°, 60°).

- *Number of energy bins:* In this experiment, the number of rays projected to each cell and the number of views (rotation angles) are fixed: 10 rays/cell and 800 views. On the other hand, the number of energy bins in which the polychromatic beam is divided takes six values: 2, 5, 10, 20, 40, and 80 bins.
- *Number of rays per cell:* The number of energy bins is fixed at 80. The number of views, at 800. The number of rays projected to each cell takes five values: 1, 4, 6, 10 and 20 rays.
- *Number of views:* The number of energy bins is fixed at 80. The rays per cell, at 800. The number of views acquired[6] takes five values: 200, 400, 600, 800, 1000.

4.3.6 Dependence on Number of Energy Bins

In order to adequately model phenomena that are dependent on the photon energy such as beam hardening, it is important to try to mimic the original distribution of photon energies by selecting a generous number of energy bins. The experiments described in this section, on the other hand, are focused mainly on the effect of varying the parameters on the spatial resolution of the system. We are therefore interested in finding what is the minimum number of energy bins that will produce simulations where the spatial resolution is not significantly affected, regardless of whether such a representation of the polychromatic beam is acceptable according to other criteria. Interestingly, the results lead us to conclude the number of bins has little effect on the

[6]The system assumes a fan angle configuration so an additional number of views is always acquired, covering at least the fan angle. The number of views mentioned in this sets of simulations correspond always to 360° of rotation and do not include these extra views.

spatial resolution. Also the differences between the images obtained and the reference images described in Section 4.3.4 are almost negligible unless the number of energy bins is less than five.

4.3.7 Dependence on Number of Rays per Detector Cell

It was also found that the number of rays projected to each detector has little effect on the spatial resolution of the reconstruction, although there is a far stronger influence when the number of rays is extremely low (less than four rays per cell). For example, in Figure 4.3 all the curves practically overlap except the one for 1 ray per cell. This behaviour is observed for all VRX angles and is totally consistent with the 5% MTF measurements (see Figure 4.4) , and, interestingly, with the differences with the corresponding reference images. The fact that the image produced when projecting only one ray per cell resulted in a slightly higher spatial resolution may seem counterintuitive, but this is actually an expected behavior because projecting only one ray to each cell is equivalent to filtering the signal received by the cell with a delta function (like using an extremely small pinhole). However this is inconsequential because in a physical detector this is equivalent to throwing out most of the photons.

4.3.8 Dependence on Number of View Angles

A much more interesting behavior is observed when varying the number of views (projection angles) in the sinogram. Figures 4.5, 4.6 and 4.7 show a clear tendency to improve when the number of views increases: better resolution is obtained, and the images get closer to their reference image. Nevertheless, for spatial resolution the advantage seems to disappear as the number of views goes over 800.

4.3.9 Conclusion

The number of rays projected to each detector cell during the simulation has little effect on the spatial resolution of the reconstruction for four or more rays. The number of energy bins used to simulate a polychromatic x-ray beam has practically no influence on the spatial resolution of the simulated image. Nevertheless, the number of bins should not be too small (around 20 to 40) in order to better model energy-dependent phenomena such as beam hardening.

On the other hand, the number of views in the sinogram does have a large influence on the resolution of the reconstructed image. This influence starts to become less noticeable when the number of views is around 800.

The comparison of the images with the corresponding reference images is in general agreement with the above conclusions.

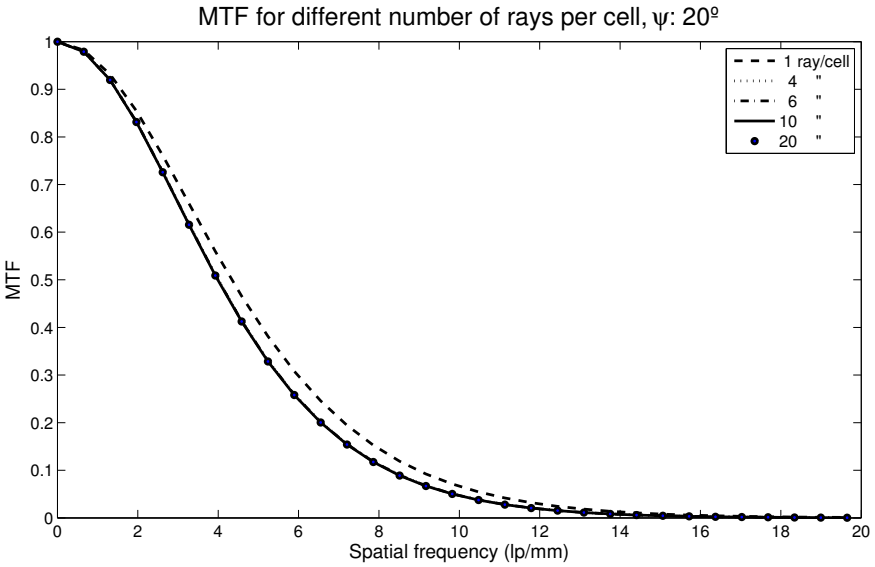


Figure 4.3: **Dependence of the MTF on the number of rays projected**. MTFs for 1, 4, 6, 10 and 20 rays projected to each cell. All but the first one are so close to each other that they seem to overlap.

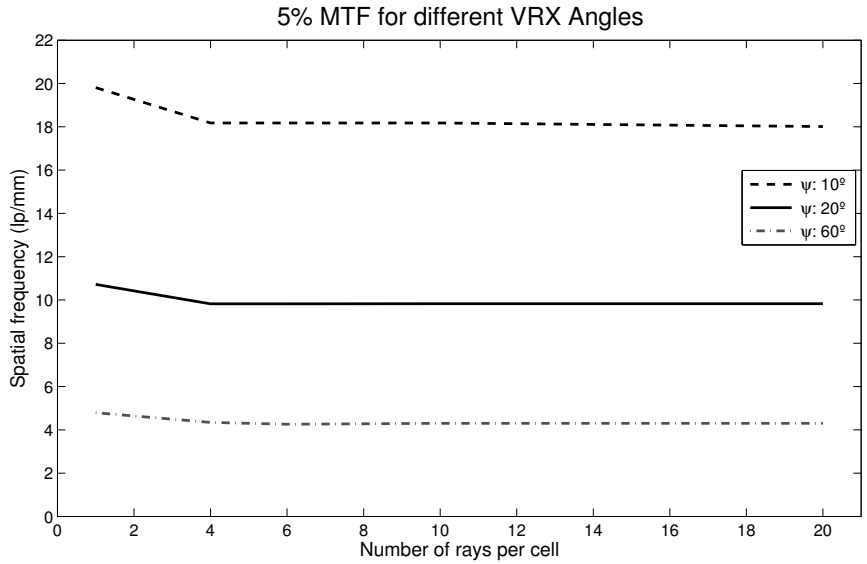


Figure 4.4: **Dependence of 5% MTF on the number of rays projected.** The 5% MTFs have almost no dependence on the number of rays per cell.

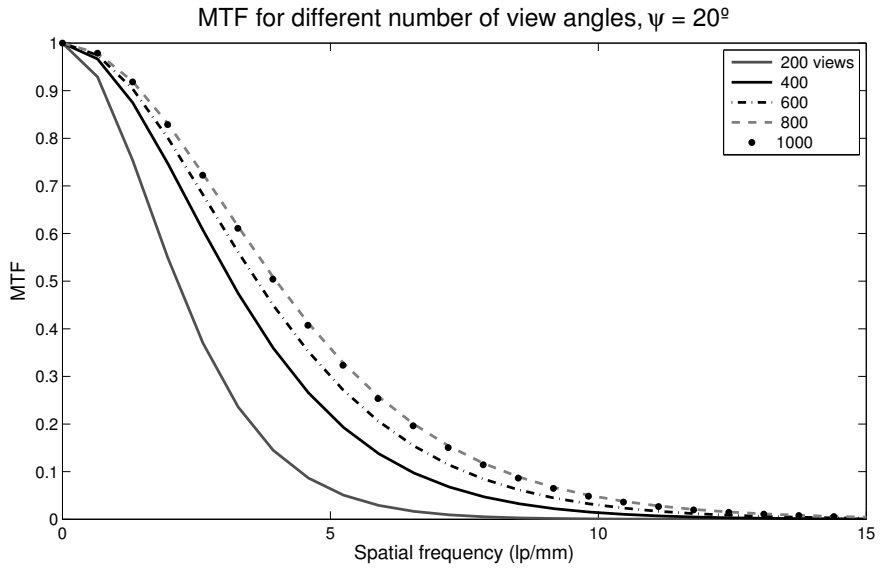


Figure 4.5: **Dependence of the MTF on the number of views.**
The MTF increases as the number of views increases. Nevertheless, at around 800 views the additional advantage seems negligible.

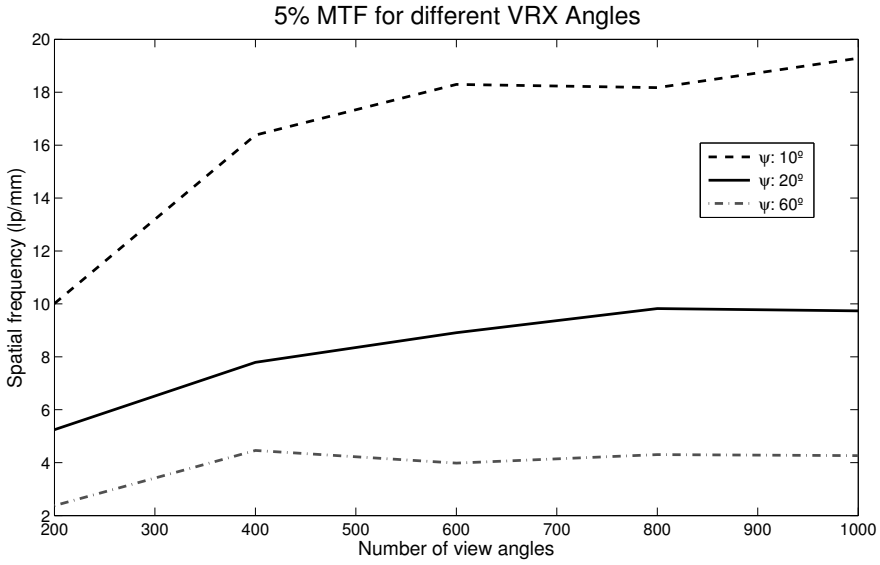


Figure 4.6: **Dependence of the 5% MTF on the number of views.**
The 5% MTFs clearly improve as the number of views increases. Nevertheless the advantage seems to taper off when a considerable amount of angles is reached.

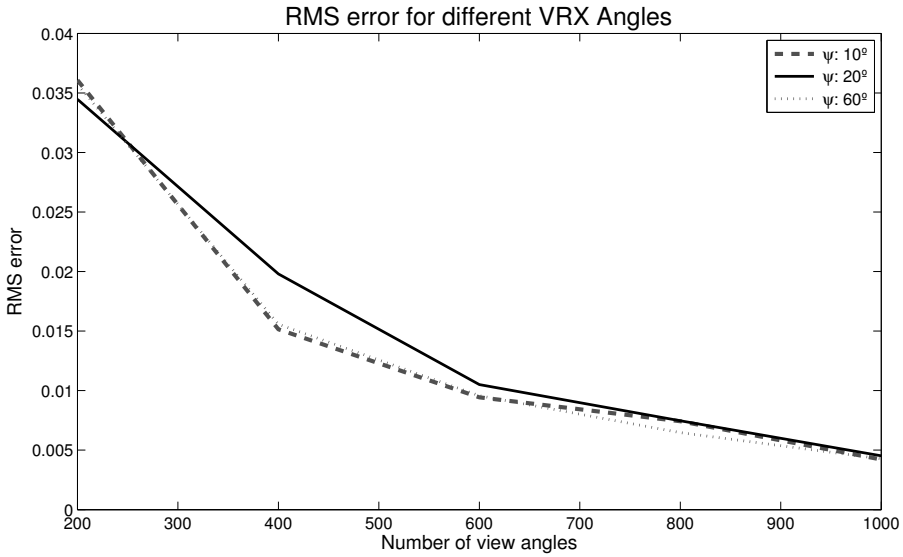


Figure 4.7: **Dependence of the difference between the virtual experiment and the reference image on the number of view angles.**
The RMS error (difference between the virtual experiment and the reference image) is clearly reduced as the number of projection (view) angles is increased.

From all this we can conclude that, under the conditions we normally use for our simulations, the best combination of parameters (i.e., the one that reduces the simulation time by using smaller parameters without compromising the adequate reconstruction of the image quality), appears to be: 20 to 40 energy bins, 4 rays/cell, 800 views per full rotation.

Chapter 5

Comparison of Targeting Scanners[1]

In Chapters 3 and 4 we introduced a set of computational tools that were developed for the study of VRX CT scanners. In this chapter and in the next one we will present two small projects that demonstrate how these tools can be extremely valuable when attempting to answer some of the questions that typically arise in the study of such devices. The project described in this chapter compares the performance of different scanner configurations that in principle appear to be roughly equivalent.

In Sections 2.3.2 and 2.3.3 we discussed how the detector of a VRX scanner can be divided into two or more separate segments, called arms, which can be placed at different angles, allowing some flexibility for the scanner design. In particular, several arms can be set at different angles creating a target region of considerably higher resolution that can be used to track the evolution of a previously diagnosed condition, while keeping the patient completely inside the field of view (FOV) [13]. The study presented in this chapter employed the computer models of single-slice VRX scanners described in Chapter 4 to examine and compare different scanner configurations (that is, various types of detectors arranged in any number of arms arranged in different geometries) in terms of spatial and contrast resolution. In particular, we were interested in comparing the performance of various geometric configurations that would otherwise be considered equivalent (using the same equipment, imaging FOVs of the same sizes, and having a similar overall scanner size), and that could be reproduced with materials readily available in our lab. Along with the VRX simulator, we developed mathematical phantoms for spatial resolution and contrast analysis.

5.1 Introduction

As described in Section 2.3, the detector of a VRX CT scanner can be divided into two or more segments, called arms, that can be tilted at different angles. Of particular interest are four-arm configurations such as the one depicted in Figure 5.1, and asymmetric two-arm configurations that resemble a letter V (pictured in Figure 5.2). In these configurations, the central region of the reconstructed image corresponds to the field of view for the arms tilted at smaller VRX angles. This region is called the

[1]Portions of this chapter adapted with permission. D. A. Rendon, F. A. DiBianca, and G. S. Keyes, "Comparison of multi-arm VRX CT scanners through computer models," *Proc. of SPIE*, Vol. 6510, San Diego, CA (2007).

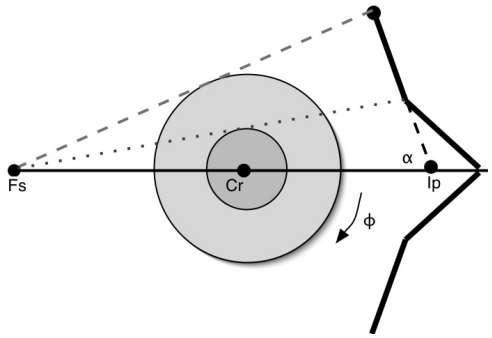
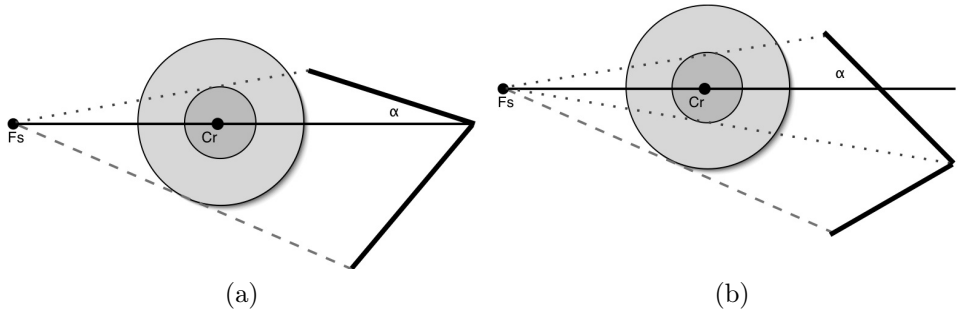


Figure 5.1: **Schematic representation of a four-arm VRX CT scanner.** (Not to scale.)



(a) (b)

Figure 5.2: **Two arm VRX CT scanners in targeting mode**. (Not to scale.) (a) Type A. (b) Type B.

target region and is imaged at a considerably higher resolution, while the *external*, lower resolution region can cover an extensive field of view. Such a target image was shown in Figure 3.7. Possible uses of target imaging include monitoring the evolution of a previously diagnosed condition, provided that the area of interest can be located at the center of the scanner.

With the development of four-arm target-imaging VRX CT scanners came the need for a special reconstruction algorithm, first presented by DiBianca et al. [13] and fully described in Section 3.1. This algorithm is independent of the number of arms and therefore can be used also for one- and two-arm VRX scanners.

The two-arm targeting systems present an additional problem, since all points in the external region will be out of the field of view of both arms at some time. To solve this problem the projection data are always acquired for all 360° of rotation, and the reconstruction software replaces the missing information at a particular angle with existing information from the low resolution arm at other angles (see Section 3.1.3). This approach adversely affects the quality of the image; in particular, the advantage of acquiring twice the information in a 360° rotation is lost for the outer region. On the other hand the quality of the image in the outer zone is not required to be as high, and these detrimental effects do not affect the target region, where the best image quality is really needed.

An interesting quality of target-imaging VRX scanners is that, given a fixed set of materials (detector arrays, etc.), it is possible to construct scanners that have a similar physical size and the same sizes for the target and outer regions, but using different configurations. For example, four linear detectors can be used to produce four- and two-arm configurations like those in Figures 5.1 and 5.2 with target and outer radii of around 1.5 and 7 cm while maintaining a similar overall size.

The study described here is mainly focused on studying three such "comparable" configurations in terms of the spatial and contrast resolutions of the images produced.

5.2 Methods

5.2.1 VRX Configurations

For this project three different multi-arm VRX configurations were studied:

- A four-arm configuration: modeled after Figure 5.1, both the inner and the outer arms were located symmetrically around the F_s to C_r axis. The inner arms formed a smaller angle with that axis, and therefore the region contained in their field of view (the target region) was imaged at a higher resolution.
- Two-arm, Type A configuration: The inner arm formed a smaller angle, and therefore its circle of reconstruction (the target region) was imaged at a higher

resolution, although that region was not always in the inner arm's field of view. See Figure 5.2(a).

- Two-arm, Type B configuration: Similar to Type A, but the target region was always in the FOV of the inner arm. See Figure 5.2(b).

All these configurations had several things in common: they used the same materials (since the longer detector arrays of the two-arm configurations can be built by attaching two of the shorter four-arm ones); they were of similar overall size; for the geometries selected, the radii of the target region and the full FOV were almost identical. Some additional details of the three configurations are summarized in Table 5.1.

5.2.2 Phantoms and Simulation Parameters

The simulator described in Chapter 4 was used to evaluate and compare images produced by the three targeting VRX configurations detailed in Section 5.2.1. In particular, two types of phantoms were defined:

- Edge phantoms, used to determine the MTF of the systems and thus evaluate their spatial resolution.
- Contrast phantoms, used to determine the SNR_C of the systems and therefore assess their contrast resolution.

5.2.2.1 Spatial resolution experiments

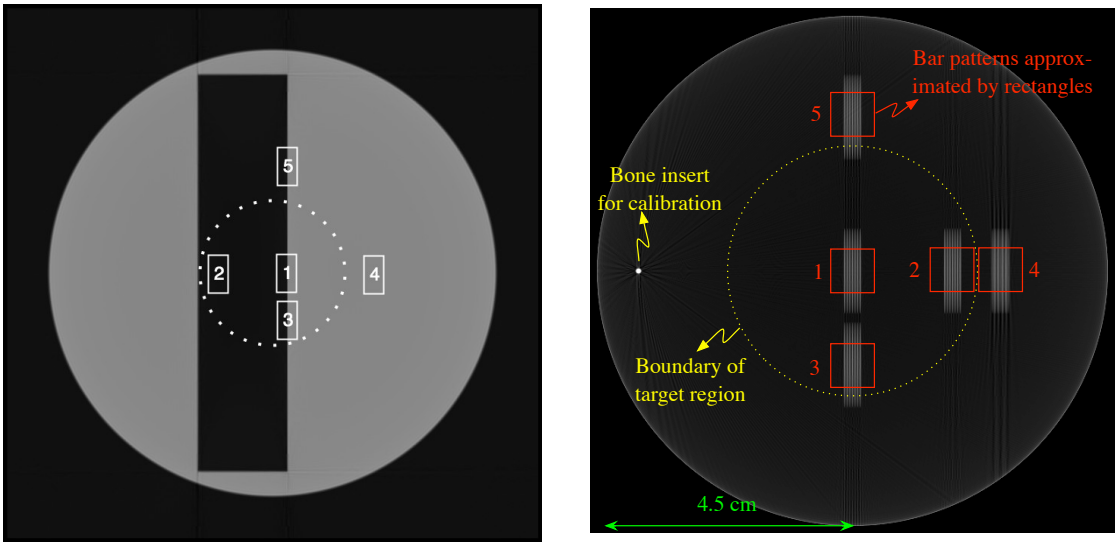
The edge phantoms used to determine the spatial resolution were defined as water cylinders with rectangular blocks of other materials inserted in them. In the results presented in this chapter, the material chosen for the inserts is air. A reconstructed image of one of these phantoms is shown in Figure 5.3(a). Note that photon noise was not added to the sinogram for the spatial resolution experiments, and thus all observed artifacts are characteristic of the device or generated by the reconstruction algorithm.

The spatial resolution in a targeting VRX scanner depends on the position in the image, as well as the orientation. As mentioned above, the resolution is considerably higher in the target zone than in the external zone, since the former is imaged with the higher resolution arms (the ones with smaller VRX angles). But points in the external zone will enter and exit the FOV of the high resolution arm(s), and thus the resolution will be different in the tangential and the radial direction, as will be evidenced in Section 5.3.1. Thus, for the analysis of the spatial resolution, we have defined five Regions of Interest (ROIs):

1. The center of the image (the central part of the target zone).

Table 5.1: **VRX configurations studied.**

Configuration	Four-arm	Two-arm Type A	Two-arm Type B
Cells per arm/total	144 / 576	288 / 576	288 / 576
Distance from C_r to detectors (along main axis)	53.0 cm	78.0 cm	53.1 cm
α (arm to axis angles)	8.00° and 32.00°	4.35° and 23.76°	9.00° and 16.81°
Radius of target region	1.46 cm	1.47cm	1.46 cm
Radius of circle of reconstruction (full FOV)	7.61 cm	7.67 cm	7.68 cm



(a) (b)

Figure 5.3: **Phantoms for spatial resolution**.
(a) Sample image of an edge phantom. The dotted line shows the approximate boundary of the target zone. The numbers correspond to the ROIs where the LSF is calculated. This phantom is used to analyze ROIs #1, 3, and 5. ROIs #2 and 4 require different phantoms. This particular image was obtained with the four-arm scanner. (b) Sample image of a high-resolution bar phantom. Note that the bar patterns are actually located regions close to the ones in (a), and that the contrast was greatly enhanced to help visualize the bars. This image was obtained with the two-arm Type A scanner.

2. The region just inside the target zone, in the radial direction.
3. The region just inside the target zone, in the tangential direction.
4. The region just outside the target zone, in the radial direction.
5. The region just outside the target zone, in the tangential direction.

The location of these regions of interest, along with the boundary of the target region, are noted in Figure 5.3(a). Additional regions farther out of the target zone should have lower resolutions, but are not of interest for this study.

For each configuration, images of edges passing through each of the five ROIs were acquired. In order to acquire a more adequate sampling [57], the virtual phantom was tilted before the simulation, as shown in Figure 5.4. All the pixels in the ROI were used to obtain a "cloud" of points that approximate the ESF, as shown in Figure 5.5. A sigmoid function was then used to fit the cloud of points, producing a smooth approximation of the ESF that was differentiated to obtain an estimate of the LSF. The LSF in turn was used to calculate the normalized MTF of the VRX configuration in that particular ROI.

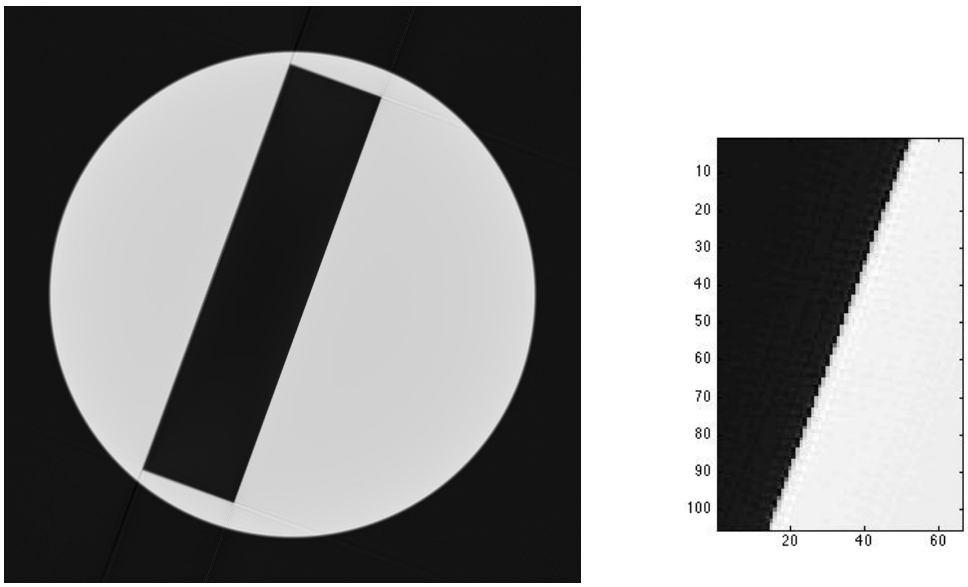
In addition to the quantitative analysis of the MTFs derived from the ESFs, these results were visually corroborated by comparing the images of high-resolution bar phantoms in the same five regions discussed above, as shown in Figure 5.3(b).

5.2.2.2 Contrast resolution experiments

The phantom defined for the contrast resolution experiments presented here was composed of a water cylinder (9 cm in diameter) with 30 cylindrical inserts of varying attenuation coefficients. The inserts were also made of water, but the density was altered, making it range from 0.91 to 1.10 g/cm^3. Therefore, some of the inserts produced positive contrast and others produced negative contrast. Two of the inserts were actually "null" inserts as they had a density of 1.0 g/cm^3. The attenuation coefficients of all the materials were obtained from the SPECT program. A reconstructed image of this phantom was shown in Figure 5.6.

The inserts were divided in two groups. The 15 larger inserts were distributed in a circle in the external region, close to the boundary with the target region, and 15 smaller ones were distributed in a smaller circle inside the target region. The purpose of this was to evaluate the contrast resolution in the external and target regions separately because, although the target region has a higher spatial resolution, the detector cells of the target arm(s) detects far fewer photons and therefore the SNR_C is lower than in the external region.

Two additional discs can be seen in Figure 5.6. These were used for calibration only. Also, that image reflects the Gaussian noise that was added to the sinogram. This was an important feature of the simulation, as SNR_C is highly dependent on the amount of noise.



(a) (b)

Figure 5.4: **Tilted edge phantom.**
(a) The ESF curves were obtained from a tilted edge phantom. (b) Rectangular ROI containing a portion of the edge.

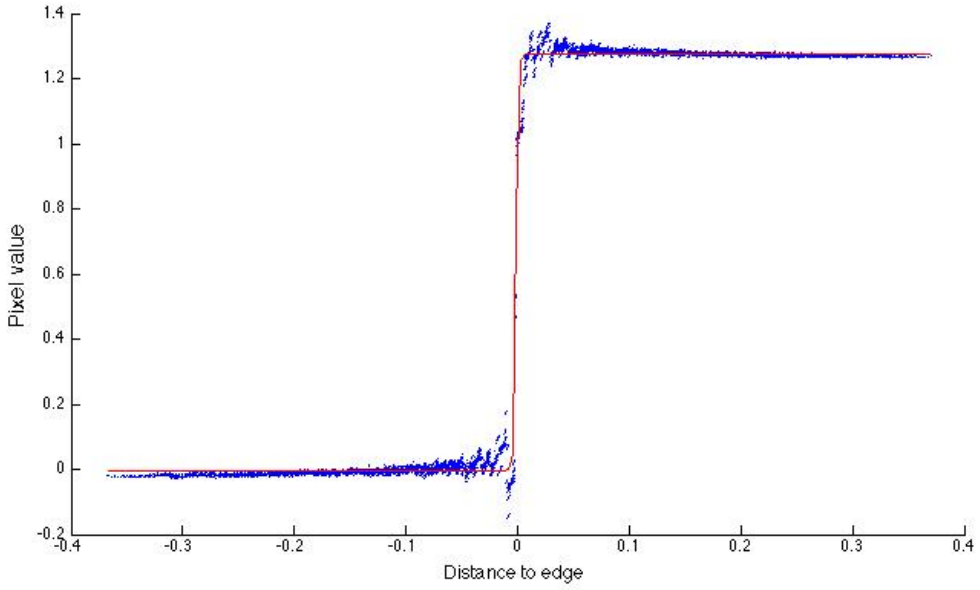


Figure 5.5: **ESF from a tilted ROI.**
Each blue dot represents a pixel from the ROI of Figure 5.4(b). The red curve is a sigmoid function fitted to the cloud, and provides a smooth approximation to the ESF.

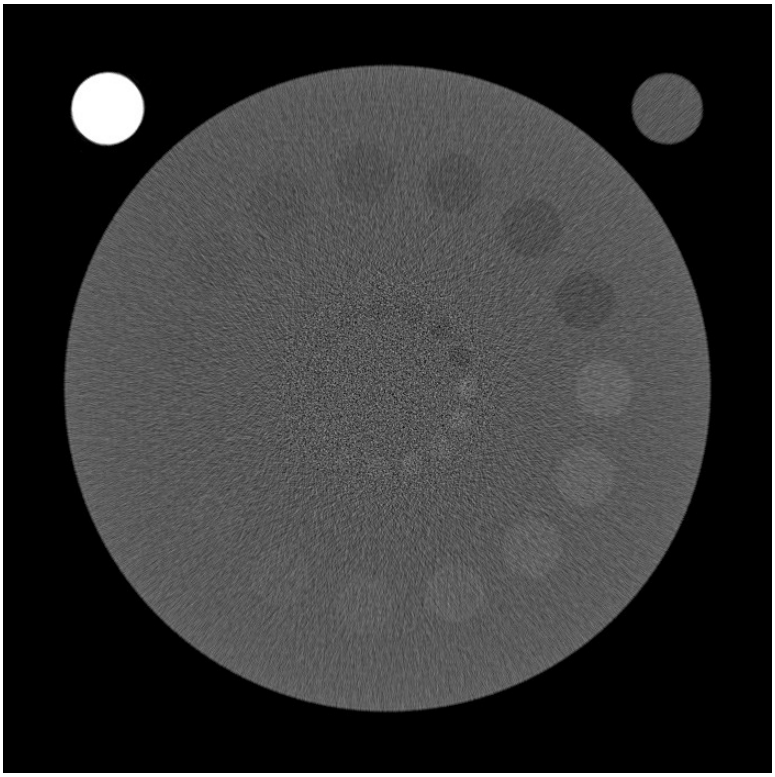


Figure 5.6: **Sample image of a contrast phantom.**
Note that the inner set of discs is located inside the target zone. This image was produced with the four-arm scanner.

A circular ROI inside each insert was defined as the signal region, and a ring outside it was defined as the background region. The pixels in these regions were used to determine the SNR_C using Equation (5.1):

$$SNR_C = \frac{\langle I_s \rangle - \langle I_b \rangle}{\sqrt{\frac{\sigma^2(I_s)}{N_s} + \frac{\sigma^2(I_b)}{N_b}}}, \qquad \text{(Eq. 5.1)}$$

where N_s and N_b are the number of pixels in the signal and background regions, $\langle I_s \rangle$ and $\langle I_b \rangle$ are the mean values of those pixels, and σ denotes the standard deviation.

5.3 Results

5.3.1 Spatial Resolution

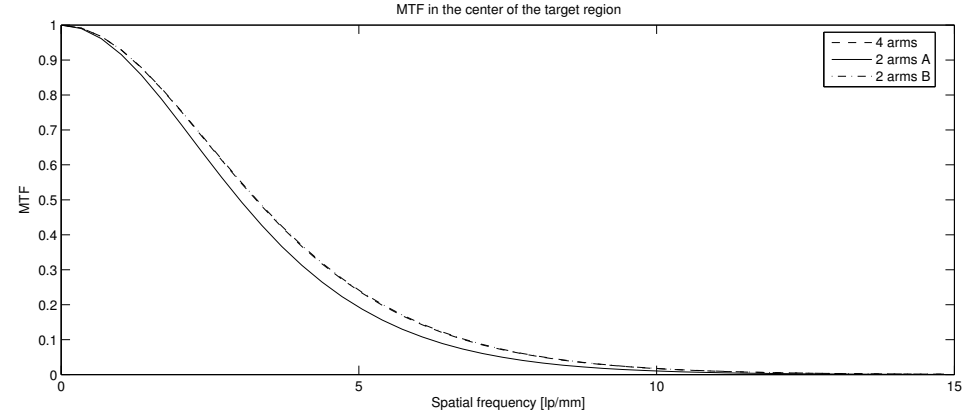
For the three VRX configurations, the LSF and MTF were determined in each of the five ROIs, as described in Section 5.2.2.1. The MTFs are shown in Figures 5.7 and 5.8. The spatial frequency at which the MTF curve first crossed the 1% threshold was considered representative of the spatial resolution for the particular combination of VRX configuration, region of the image, and orientation (radial or tangential). Those values appear in Table 5.2.

From these results we can see that the three configurations produced comparable results near the center of the target region. As we move towards the target region's boundary, we start to see an effect of the orientation in which we measure the LSF. In the radial direction the two-arm Type B configuration had a considerably higher MTF, as well as a much higher cut-off frequency. In the tangential direction all the systems performed similarly.

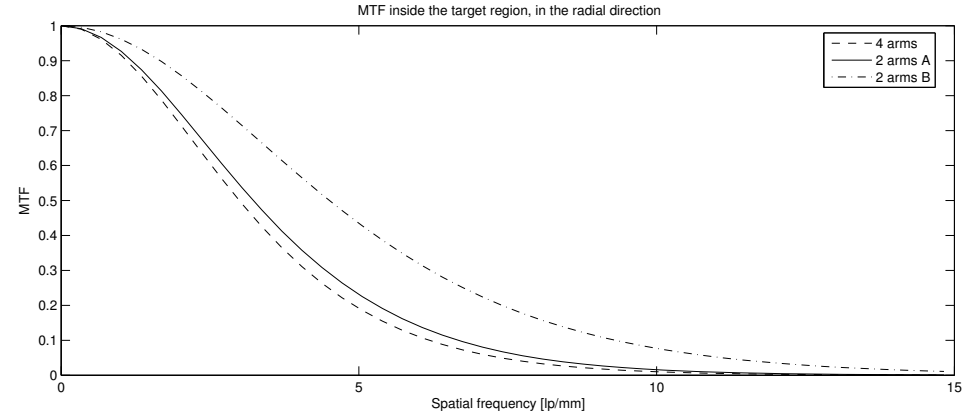
In the external region we found an interesting phenomenon: the MTFs in the tangential direction did get reduced (as expected) when compared to the target region, but the overall effect was not significant. This can be explained by the fact that the resolution in the tangential direction for the outer region is affected mainly by backprojections originated in the target arm(s). One should keep in mind that all points in the external region go in and out of the line of sight of the target arm(s).

The behavior of the MTF curve in the radial direction was expected to be worse, since it got more contribution from backprojections originated in the low resolution arms, something confirmed by the experiments. While the two-arm Type A configuration surprisingly performed relatively well, the other two configurations performed poorly. In particular the four-arm configuration had the lowest performance of all the ROIs studied.

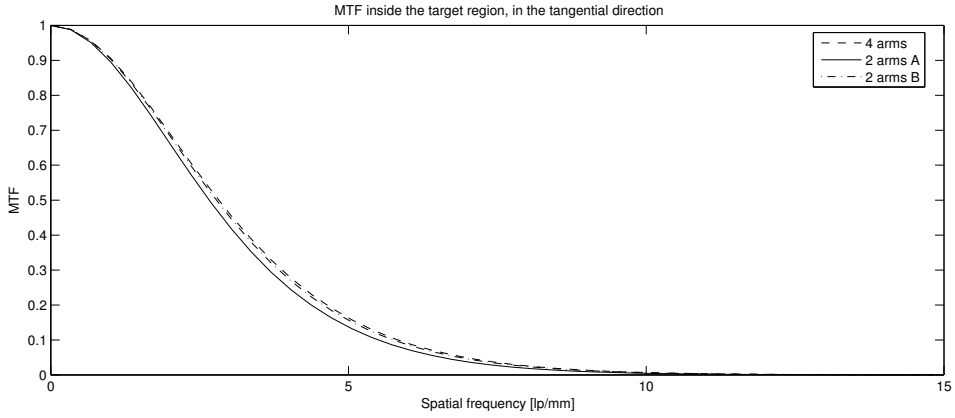
Figure 5.9 provides a visual corroboration of the results for the spatial resolution, by comparison of the spatial resolutions at the five regions considered. The bar



(a)

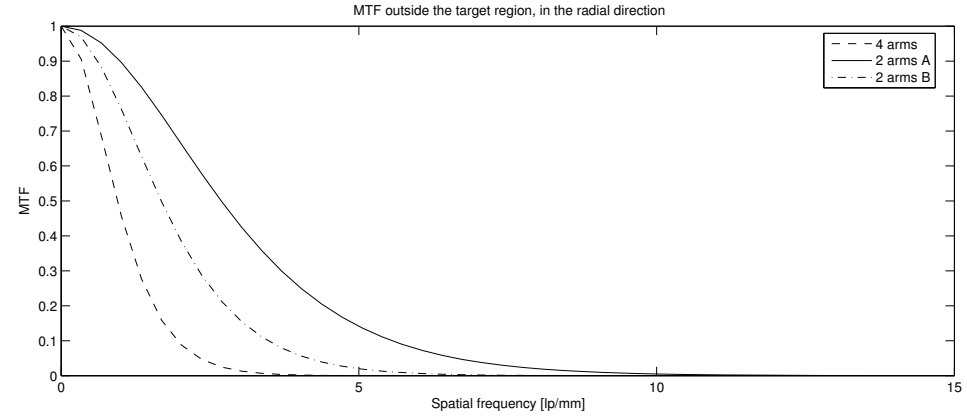


(b)

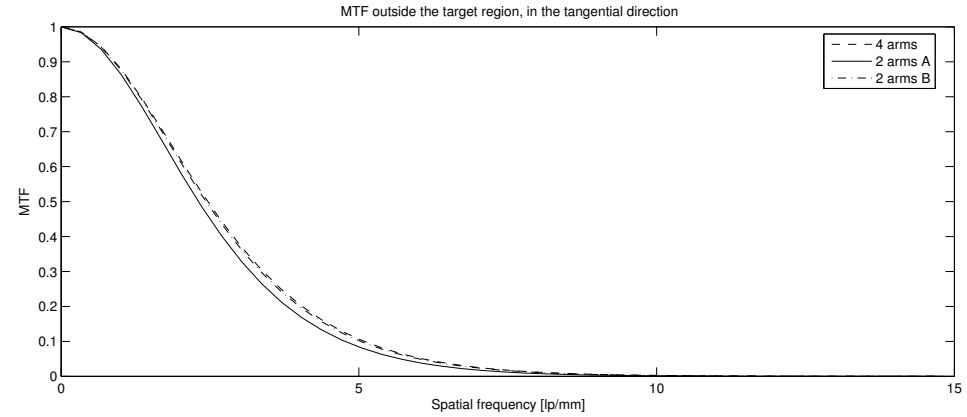


(c)

Figure 5.7: **MTF for the three systems inside the target region.** (a) Region 1. (b) Region 2. (c) Region 3.



(a)



(b)

Figure 5.8: **MTF for the three systems outside the target region.**
(a) Region 4. (b) Region 5.

Table 5.2: **Spatial resolutions in the five ROIs for each VRX configuration, in lp/mm.**

#	Region	Four-arm	Two-arm Type A	Two-arm Type B
1	Central	10.9	10.0	11.0
2	Target - radial	10.0	10.8	15.0
3	Target - tangential	9.4	8.9	9.3
4	External - radial	3.2	9.0	5.6
5	External - tangential	8.2	7.7	8.1

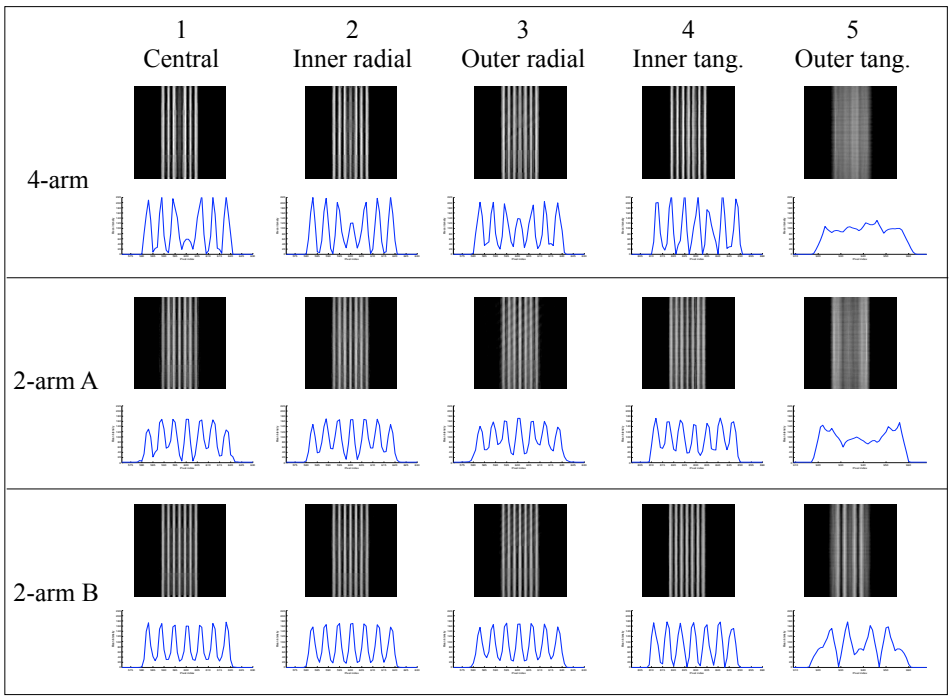


Figure 5.9: **Comparison of the spatial resolutions at the five regions considered**.
The bar patterns have a spatial resolution of 3.33 lp/mm.

patterns have a spatial resolution of 3.33 lp/mm, so the three scanners are easily able to image them anywhere in the target region. In the outer region, the resolution is fairly acceptable in the radial direction, but severely degraded in the tangential direction. Note that the central bar for the four-arm configuration appears highly attenuated; this is caused by a small artificial gap introduced in the vertex to better approximate the geometry of the physical scanner used to acquire the forearm image shown in Figure 3.7.

5.3.2 Contrast Resolution

The phantom shown in Figure 5.6 was used to evaluate the contrast resolution of the three configurations in both the target and the external region. The results for the SNR_C are shown in Tables 5.3 and 5.4, and in Figure 5.10.

In general all three systems performed similarly, with a small advantage for the two-arm Type B configuration in the target region (it had slightly higher absolute SNR_C). Nevertheless, the SNR_C values for the target region varied considerably, undermining the significance of this conclusion. This variability was partially due to the lower number of photons detected by the high resolution arm, and also to the fact that the we needed to use smaller ROIs, which adversely affects the statistical significance of the measurements in that region. In the external region the behavior was much more linear, as expected. Here the Type A system had a small disadvantage.

5.4 Conclusions

The spatial resolution of the targeting VRX CT scanners modeled in this study was extremely good. In reality the size of the focal spot of the x-ray tube (which was not included in the computational model) has a large impact on the final image quality. Also, while the VRX angle of the arms helped increase the spatial resolution, the detection of photons was decreased, thus affecting negatively the contrast resolution, as confirmed by the experiments. This was specially evident in the target region.

It is interesting to see the effect that the orientation had on the spatial resolution in the external region. As discussed in the results, this was due to the fact that all points in that region enter and leave the field of view of the target arm(s), and thus some of the backprojections that make up the reconstruction are inherently of higher resolution than others.

The results show that in general the two-arm Type B configuration had a small advantage over the other two in terms of both spatial and contrast resolution. This was true in all the different regions of the image except for the external region in the radial direction. It is surprising that the Type A configuration excelled in that situation, due to the fact that for that configuration even the points in the target region leave the FOV of the target arm at some point of the scan. It must be noted

Table 5.3: **SNR$_C$ in the target zone for each VRX configuration**.

ρ	Four-arm	Two-arm Type A	Two-arm Type B
0.909	−6.96	−6.03	−9.11
0.922	−5.44	−4.60	−7.39
0.934	−6.00	−6.04	−5.67
0.947	−2.85	−3.02	−5.22
0.960	−2.64	−2.52	−4.61
0.973	−2.67	−2.15	−1.53
0.987	−2.11	−1.46	−1.60
1.000	−1.86	−0.81	1.51
1.014	1.43	1.60	2.51
1.028	1.54	1.41	2.22
1.042	3.68	4.11	3.73
1.056	4.67	3.18	3.63
1.070	4.01	6.04	4.87
1.085	4.53	5.66	7.99
1.100	6.58	6.82	8.86

Table 5.4: **SNR$_C$ in the external zone for each VRX configuration**.

ρ	Four-arm	Two-arm Type A	Two-arm Type B
0.909	−36.99	−33.25	−39.35
0.922	−34.96	−29.48	−33.09
0.9342	−27.80	−27.63	−34.86
0.947	−24.03	−20.27	−24.31
0.960	−17.21	−13.84	−13.70
0.973	−12.21	−8.27	−11.18
0.987	−5.21	−4.45	−7.16
1.000	−1.13	−0.77	1.58
1.014	7.03	3.72	4.69
1.028	10.66	7.79	13.28
1.042	15.72	16.98	17.52
1.056	25.55	18.84	25.08
1.070	27.47	22.79	29.55
1.085	33.98	30.68	34.59
1.100	40.95	35.66	41.07

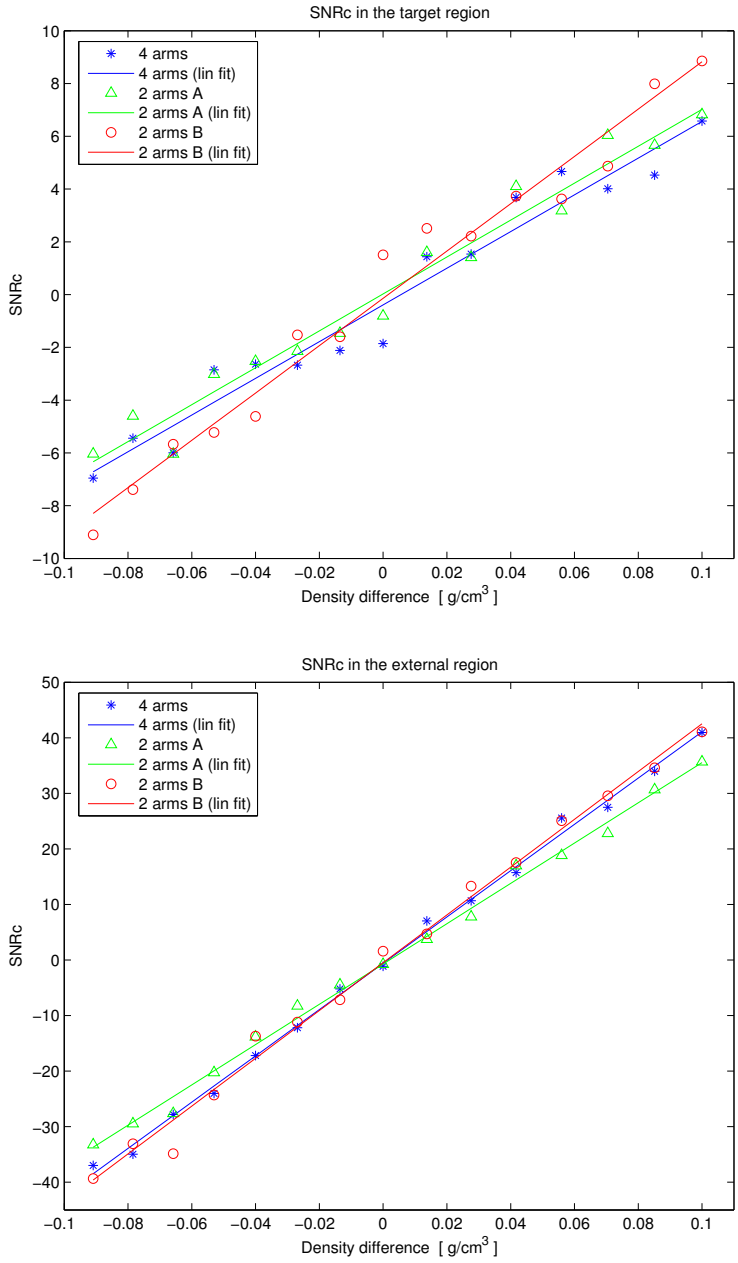


Figure 5.10: **Comparison of SNR_C for the three systems**.
(a) In the target region. (b) In the external region.

that this configuration had the worst performance in almost all the other ROIs. The poor performance of the four-arm scanner in the external region, especially in the radial direction, came as a surprise and may deserve closer attention. In general we must conclude that, even though intuitively the four-arm system could have a small advantage due to its symmetry, in reality the two-arm Type B configuration exhibited the most consistent and superior results.

Chapter 6

Artifacts in Images of VRX Scanners[1]

As with regular CT scanners, the images obtained with VRX scanners are affected by different kinds of artifacts of various origins. The work presented in this chapter studied some of these artifacts and the impact that the VRX effect has on them. For this, computational models of single-arm single-slice VRX scanners were used to produce images with artifacts commonly found in routine use. These images and artifacts were produced using the VRX CT scanner simulator described in Chapter 4, which allowed us to isolate the system parameters that have a greater effect on the artifacts. A study of the behavior of the artifacts at varying VRX opening angles was presented for scanners implemented using two different detectors. The results show that, although varying the VRX angle will have an effect on the severity of each of the artifacts studied, for some of these artifacts the effect of other factors (such as the distribution of the detector cells and the position of the phantom in the reconstruction grid) is overwhelmingly more significant. This is shown to be the case for streak artifacts produced by thin metallic objects. For some artifacts related to beam hardening, their severity was found to decrease along with the VRX angle. These observations allow us to infer that in regular use the effect of the VRX angle on artifacts similar to the ones studied here may not be noticeable as it may be overshadowed by parameters that cannot be easily controlled outside of a computational model. Hence this is an example of how the use of a virtual scanner may preclude the need to conduct cumbersome and time-consuming experiments which are unlikely to show the desired results.

6.1 Introduction

As with regular CT scanners, the images obtained with VRX scanners are affected by different kinds of artifacts. Some artifacts are due to phenomena inherent to x-ray imaging such as beam hardening. Others are specific to the x-ray detector technology used. Still others are due to the image acquisition protocol and the reconstruction algorithm [6]. The work described in this chapter studied some of these artifacts and the impact that the VRX effect has on their severity. For this, compu-

[1]Portions of this chapter adapted with permission. D. A. Rendon, F. A. DiBianca, and G. S. Keyes, "Reconstruction artifacts in VRX CT scanner images," *Proc. of SPIE*, Vol. 6913, San Diego, CA (2008).

tational models of single-arm single-slice VRX scanners were used to produce images with artifacts commonly found in routine use, such as streak artifacts produced by highly absorbent objects such as bones, needles, and calibration pins. The images were then evaluated to assess the severity of the artifacts.

The images were produced by a computer model of single-slice VRX scanners described in Chapter 4. Two different single-arm detectors were modeled to determine if the distribution of the cells in the arm had an influence on the results. For both detectors, images were obtained with the arm located at several different VRX angles, ranging from 10° to 90°. A schematic representation of the setup is shown in Figure 6.1.

The images were produced using two types of phantoms. In one type, a thin and long metallic object was immersed in a disc of water. The metallic object can represent a biopsy needle entering the phantom through the side, or a cross-section of a metallic foil. Such an object will produce huge streak artifacts along its main axis, as shown in Figure 6.2. In the other type of phantom, two elliptical objects of bone material were located in a cross-section of muscle and fat, as shown in Figure 6.3(a). In such a configuration, beam hardening artifacts produced by both bones will interact producing a very characteristic pattern that can be better observed in Figure 6.3(b).

The evaluation of the severity of the artifacts was done by comparing the image with a map of the expected materials for each pixel. Because the phantoms were described by mathematical formulas, for each pixel in the image it was possible to determine the materials that composed it and therefore the expected value of the pixel. A region of interest (ROI) that contained the artifact was selected, and for each pixel in it the error for that pixel was given by the difference between the expected and the obtained values. A global measure of the severity of the artifact was obtained by determining a distance between the two ROIs. In our case, we used the root mean square of the pixel errors as the global measure of severity.

6.2 Methods

6.2.1 Computational Models of VRX Scanners

Computational models are extremely useful for the development and research of CT scanners since they allow us to explore and compare different configurations without necessarily building them all. Through simulations it is also possible to isolate the effect produced in the final image by each one of the many elements that interact in the system [51]. In this study, these two advantages of the computer simulator of VRX scanners were fully exploited. The simulator allowed us to contrast the results obtained by running the experiments with two different detectors. It also allowed us to minimize or eliminate effects that would affect our measurements

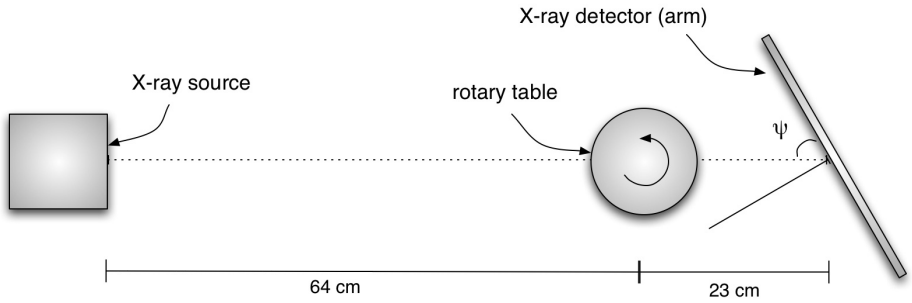
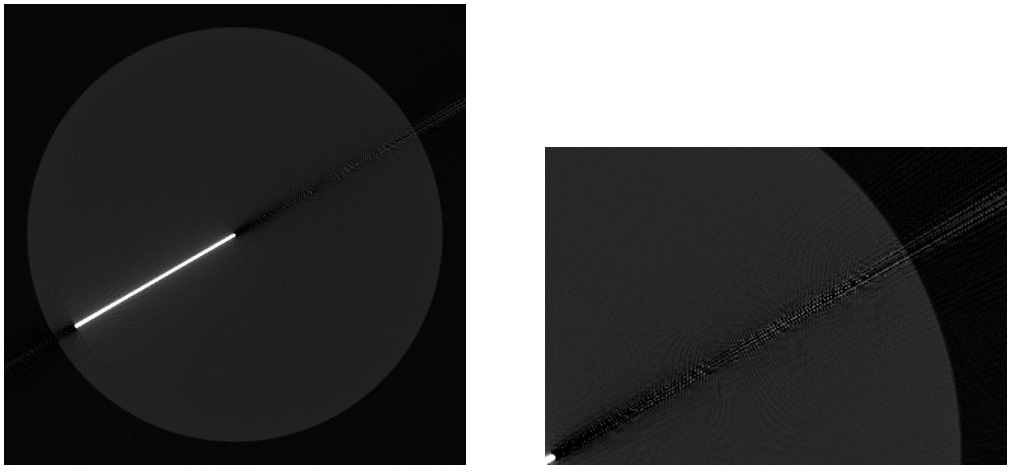
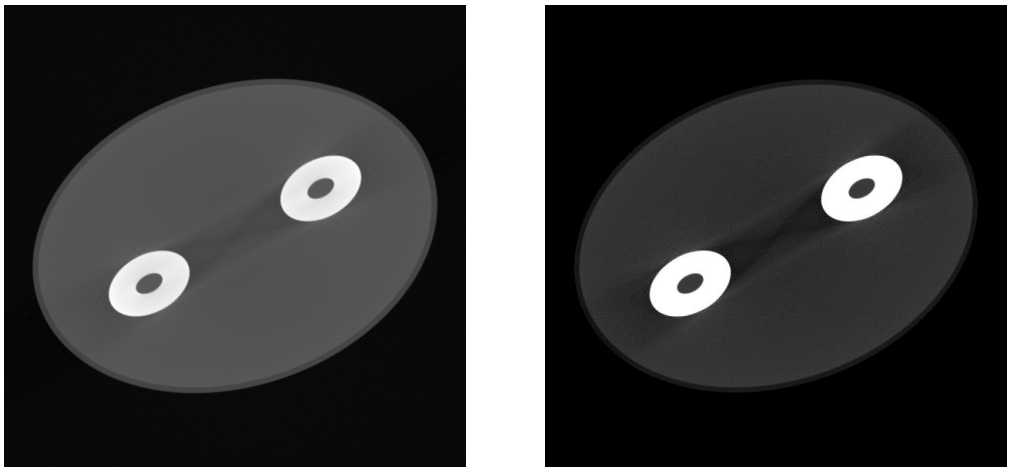


Figure 6.1: **Schematic representation of the simulated VRX CT setup**. ψ is the VRX angle.



(a) (b)

Figure 6.2: **Streak artifacts produced by a long, thin metallic object**. (a) Image of the complete water phantom with a thin metallic insert. (b) Detail of the area containing the streak.



(a) (b)

Figure 6.3: **Beam hardening artifacts between two bones.**
(a) Beam hardening artifacts produce a particular pattern between two bones in a cross-section of muscle and fat tissue. (b) The same image, with the window width and level adjusted to enhance the artifacts.

without providing any valuable information. For example, no noise was added to the sinograms because the artifacts we were studying were not produced by noise, and including noise would diminish our ability to acquire the measurements accurately and repeatably. Finally, the extremely accurate description of all the elements in the system, coupled with the ability to perform several experiments (even in parallel) without the overhead of preparing each physical experiment, allowed us to perform many hundreds of measurements under various conditions. This included numerous combinations of VRX angles and positions of the artifact-generating objects.

All the virtual experiments for this project were performed assuming the same geometrical arrangement of the parts (see Figure 6.1). For the x-ray source, a polychromatic beam (120 kVp, 100 mA, 3 mm Al intrinsic filtration) originating from a focal spot of negligible size was assumed. The initial energy distribution of the beam and the attenuation coefficients of the materials involved were obtained from the "SPECT" (x-ray spectrum) program, as described in Chapter 4. In order to simplify the analysis, a single-arm VRX scanner was modeled.

Two different detectors were modeled to examine the dependence of the severity of the artifacts on the physical distribution of detector cells on the arm. The detectors modeled are an X-Scan f2-307 HE (linear array detector with $CdWO_4$ scintillators, Detection Technology, Inc., Ii, Finland) and a PaxScan 2020 (amorphous Si flat panel detector with CsI scintillator, Varian Medical Systems, Salt Lake City, UT, USA). Only one row of cells from the PaxScan was modeled in the single-slice scanner. Table 6.1 shows the most relevant parameters of both detectors along with the length of the side of the reconstructed region at some representative VRX angles (the complete fields of view are not shown in the figures included here, only the central region). In each complete set of experiments, the VRX angle took all the values between 10° and 90°, in intervals of 2.5°.

6.2.2 Phantoms and Artifacts

This study focused on two types of artifacts, both produced by objects with high absorptivity. In the simulations these artifacts were replicated by phantoms described by mathematical formulas in which each object was identified with a material for which the polychromatic attenuation coefficients were obtained from the SPECT program.

In the course of this study it became evident that the exact relation between the objects in the phantom and the reconstruction grid had a much bigger impact on the severity of the artifacts than the VRX angle of the detector, especially for streak artifacts (see Section 6.3). This presents a major problem if the same phantom is used to generate images at different VRX angles, as the reconstruction grid will adapt to the varying FOV causing variations in the artifacts that will overshadow the effect of the VRX angle. Since the main goal of this project was to analyze the effect of the VRX angle, it was necessary to create phantoms that would adapt to the

Table 6.1: **Size of the reconstructed region at various VRX angles**.

			Size (in cm) at VRX angle			
Device	# of cells	Pixel pitch	10°	30°	50°	90°
PaxScan 2020	1024	194 μm	1.72	4.88	7.25	8.74
X-Scan f2	768	400 μm	3.10	8.65	12.52	14.30

reconstruction grid. This was achieved by creating, for each VRX angle, a phantom that was scaled to match the corresponding FOV.

Figure 6.2 shows an image of a phantom that produces noticeable streaks along the main axis of a long and thin metallic object immersed in water. Images with streak artifacts like these are frequently produced by objects such as biopsy needles entering the phantom through the side, or metallic foils perpendicular to the imaging plane. We therefore refer to this phantom as a *needle phantom*, although it could better represent the cross-section of a foil. Table 6.2 shows the parameters used to generate these phantoms. In order to evaluate the influence of the position of the needle with respect to the reconstruction grid, all the experiments were repeated with the needle forming different angles with the horizontal axis. The values of these angles are shown in Table 6.2. Note that these angles include 30° and values of $30° \pm 2^n$, with n taking all integer values from -3 to 3.

Figure 6.3 shows images of a phantom that represents two highly absorbent objects that produce artifacts that interact forming a characteristic pattern. This kind of pattern is sometimes seen in images of regions that contain several bones, such as the forearm and parts of the skull, and are due to beam hardening [6]. We refer to this phantom as a *bone phantom.* Table 6.3 shows some of the parameters used to generate these phantoms. As with the needle phantom, the experiments were repeated with the phantom rotated at small angles to evaluate the influence of the position of the objects with respect to the image grid.

6.2.3 Ideal Images

The severity of the artifacts was quantified in this study by measuring how much the reconstructed image deviated from an "ideal" or "desired" image. Each of the objects in the phantoms was associated with a particular material, and was described by a combination of mathematical formulas; therefore, each pixel in the reconstructed image can be associated with the ideal value for the corresponding material.[2] This value should be proportional to the attenuation coefficient of the material at the

[2]A pixel can, in fact, be composed of more than one material (e.g., on the interface between objects). To address such cases, the ideal image is constructed on a grid with twice the resolution of the reconstructed image, and then scaled back through bilinear interpolation.

Table 6.2: **Parameters for the needle phantoms**.

Parameter	Value
Needle material	Iron
Background disk material	Water
Diameter of background disk	0.9
Length of needle	0.4
Thickness of needle	0.005
Angle of the needle (degrees)	22, 26, 28, 29, 29.5, 29.75, 29.875, 30, 30.125, 30.25, 30.5, 31, 32, 34, 38

Notes: Lengths are given as fractions of the side of the reconstructed region (see Table 6.1 for some representative values). One side of the needle is located at the center of the FOV. The angles of the needle are measured with respect to the horizontal axis.

Table 6.3: **Parameters for the bone phantoms**.

Parameter	Value
Materials used	Skeletal bone, muscle, fat
Background tissue	Muscle tissue (0.875×0.625) surrounded by 0.025 units of fat
Bones	0.1875×0.125, with "marrow" of size 0.0625×0.0375
Distance between bone centers	0.425
Angle of the axis formed by the bones (degrees)	22, 26, 28, 29, 29.5, 29.75, 29.875, 30, 30.125, 30.25, 30.5, 31, 32, 34, 38

Notes: Lengths are given as fractions of the side of the reconstructed region (see Table 6.1 for some representative values). The angles of the bones axis are measured with respect to the horizontal axis.

effective energy of the scanner. Unfortunately, determining the most adequate value analytically is a cumbersome task due to different phenomena (including hardening of the x-ray beam, miscalibration of the scanner, etc.) [36].

A simpler approach was followed in this study: for each experimental image we produced a *calibration phantom* that contained all the materials involved and that had a size comparable to the mathematical phantom used in the virtual experiment. This calibration phantom was imaged using the exact parameters used to produce the experimental image, and a reasonable average value was then obtained from the reconstructed calibration image for each of the materials. A typical calibration image is shown in Figure 6.4(a), and an ideal image created using values derived from it is shown in Figure 6.4(b). After analysing many images produced this way we concluded that, as expected, the pixel values of the ideal image were good approximations of the values of the corresponding pixels of the experimental images in regions *devoid* of artifacts. It is therefore reasonable to use these values as the desired or expected values in regions that are obscured by artifacts.

6.2.4 Comparison of Simulated and Ideal Images

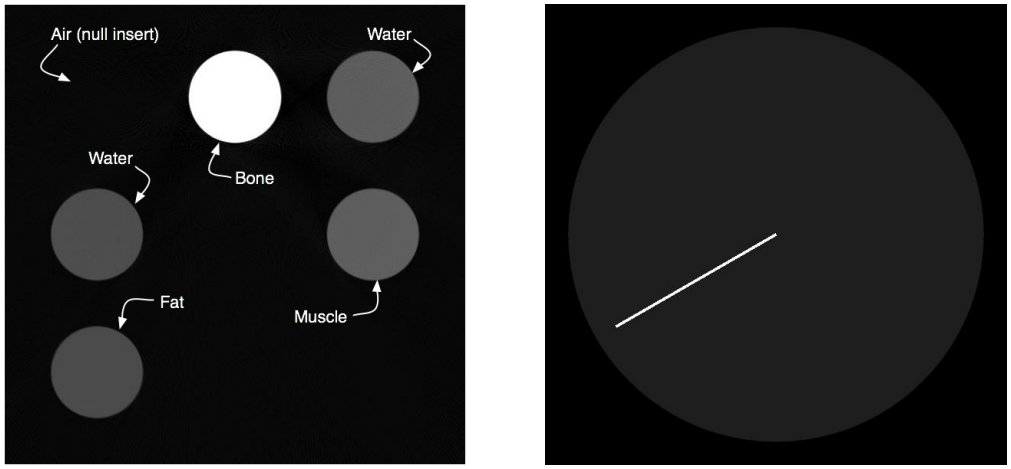
In order to calculate the error between the ideal and the experimental image due to the artifacts, we defined a ROI that isolates the artifact being studied. Only pixels within the ROI were be used in the calculation of the error introduced by the artifacts of interest.

In the case of images of the "needle" phantoms such as Figure 6.2, we were interested in the streaks generated in the upper right-hand quadrant of the image. The ROI chosen was delimited by two lines that formed an angle of 4°, and intersected just inside the needle so the streaks (but not much unassociated background) would be completely contained. Figure 6.5 shows a binary mask representing this ROI and streaks extracted by applying this mask to a simulated image. For the "bone" phantoms, the ROI chosen was a rectangle inscribed between the two bones, and slightly taller than them. Figure 6.6 shows a mask representing such a ROI and artifacts extracted using it.

For each pixel, the error was defined as the difference between the ideal value and the one obtained. For a measure of the total error we chose the root mean square of the errors for all the pixels in the ROI:

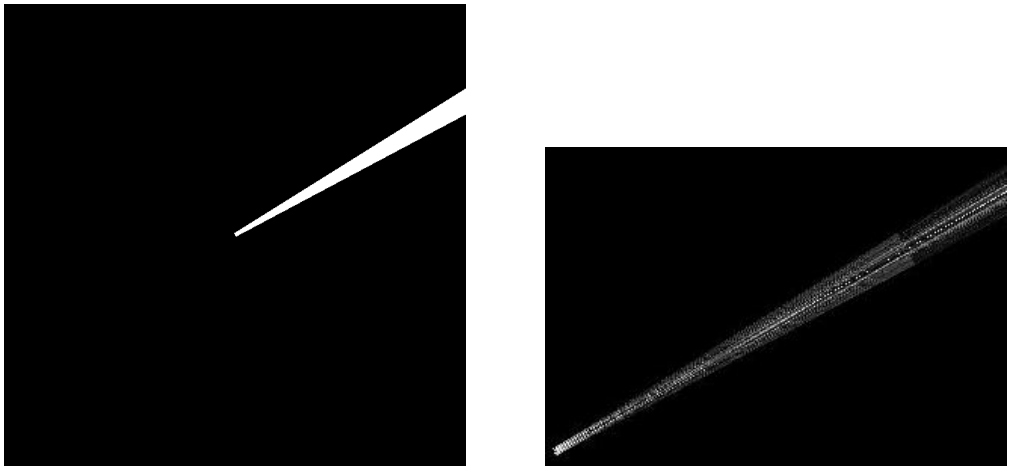
$$E_{\text{total}} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{j,\text{sim}} - x_{j,\text{ideal}})^2},$$

where $x_{j,\text{sim}}$ and $x_{j,\text{ideal}}$ are pixel values obtained from the simulation and from the ideal image, respectively, and n is the number of pixels in the ROI.



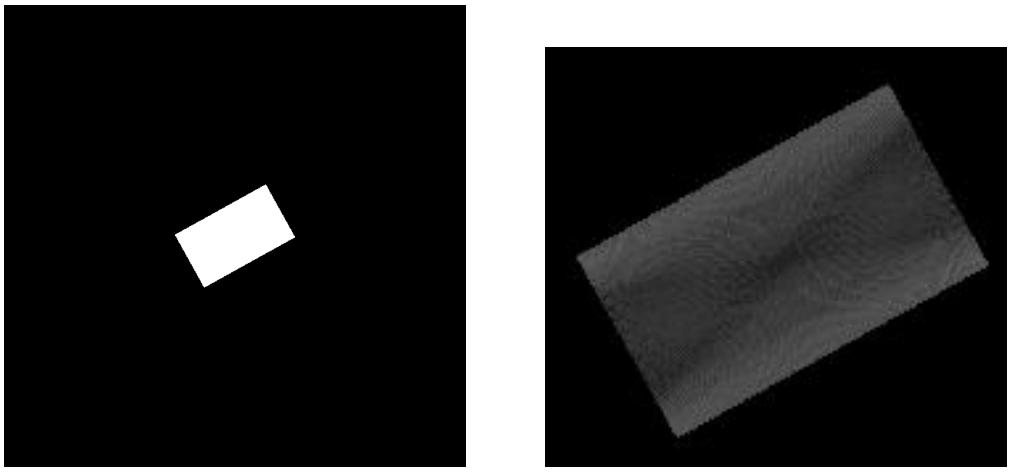
(a) (b)

Figure 6.4: **Calibration phantom and "ideal" image**.
(a) Example of a calibration phantom. (b) "Ideal" image corresponding to Figure 6.2 generated using the values obtained from a calibration phantom.



(a) (b)

Figure 6.5: **Binary masks for the extraction of streak artifacts**.
(a) Binary mask that specifies a ROI that contains the streak artifacts of Figure 6.2.
(b) The streaks extracted from Figure 6.2 by applying the mask. (Detail)



(a) (b)

Figure 6.6: **Binary masks for the extraction of beam hardening artifacts.** (a) Binary mask that specifies a ROI that contains the main beam hardening artifacts of Figure 6.3. (b) The streaks extracted from Figure 6.3 by applying the mask. (Detail)

6.3 Results and Discussion

The process of obtaining a simulated image with artifacts, a calibration phantom, an ideal image and a ROI mask was repeated for all combinations of VRX angles between 10° and 90° in intervals of 2.5° and positions of the phantom (the angles described in Tables 6.2 and 6.3). For brevity, only a selection of the most significant results are presented here.

6.3.1 Streak Artifacts Produced by the "Needle" Phantom

Figure 6.7 shows the RMS error introduced by the streak artifacts in the needle images when the needle formed an angle of 30° as a function of the VRX angle, for both detectors. All the curves for the PaxScan follow a similar pattern to the one observed in the figure, and this is also the case for most (13 out of 15) curves for the X-Scan. Several things in this plot are noteworthy:

- Both curves appear to consist of two segments. This was observed in nearly all the experiments and we believe it may be linked to the fact that the improvements in resolution provided by VRX scanners become substantial enough to overcome the deleterious effects of detector tilt only for angles smaller than a threshold which depends on the specific detector architecture.
- The errors for the PaxScan are greater than for the X-Scan.
- The errors for the PaxScan generally increase as the VRX angle increases (except for a very small dip in the middle and, in some cases, for angles greater than 80°), while they generally decrease for the X-Scan (except for a cusp that is typically found around a VRX angle of 57.5°).

Figure 6.8 shows additional curves for both devices, corresponding to other needle angles. Although the curves tend to follow similar patterns, it is interesting to note that the order of the curves is in no way related to the needle angle. In fact, plotting the RMS error as a function of the needle angle for fixed VRX angles produces completely disjointed curves.

The results showcased in Figures 6.7 and 6.8 suggest that, even though the VRX angle does have an influence on the severity of the artifacts, the exact nature of the relation is far from clear. Furthermore, moving the object that generates the artifacts slightly (for example, rotating it by 1/8 of a degree), will cause a significant and seemingly erratic change in the severity of the streaks. This means that in a physical scanner the effect of the VRX angle on the streaks would be almost impossible to measure: it is extremely difficult to place the phantom in an exact position, and a minuscule error would produce effects comparable to the variations that are being measured.

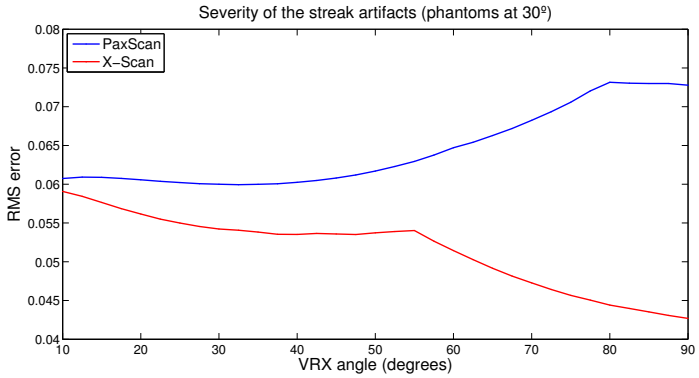
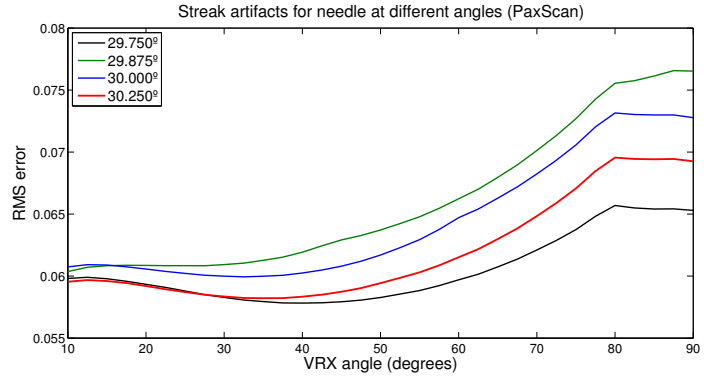
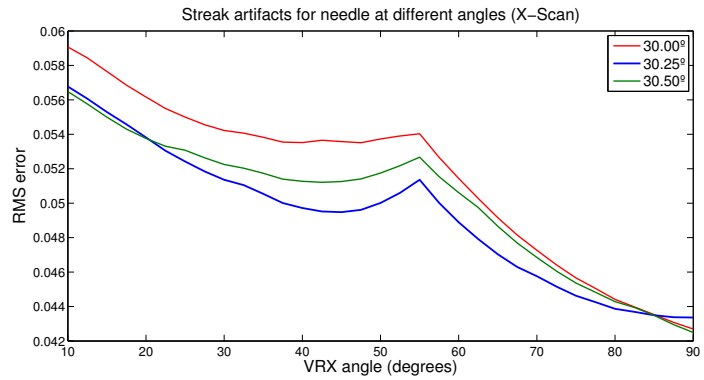


Figure 6.7: **Severity of the streak artifacts as a function of the VRX angle for both detectors**.
The needle is located at 30° in both cases.



(a)



(b)

Figure 6.8: **Severity of the streak artifacts for different needle angles**.
(a) For PaxScan 2020. (b) For X-Scan f2-307.

6.3.2 Beam Hardening Artifacts in the Bone Phantoms

In contrast to the mixed results obtained for the streak artifacts, the artifacts produced by the bone phantoms had a clear relationship with the VRX angle. Figure 6.9 shows representative results for both detectors. The four curves (two for each device) correspond to different angles formed by the main axis of the bone structures and the horizontal axis. All the curves that were omitted (which correspond to angles ranging from 22° to 38°) fall within the ones depicted for the corresponding device, showing a consistent behavior.

The direct relation of the artifact severity with the VRX angle may seem surprising at first, but it is in fact expected due to the protocol followed: for each VRX angle we are constructing a phantom proportional to the FOV; thus, for smaller VRX angles the "bones" in the phantom will be smaller and therefore affect the quality of the beam to a lesser degree.

It remains to be shown what the results would be if the same phantom (of fixed size) had been used for all the VRX angles. On the other hand, such an experiment would imply that either the phantom is extremely small (so that it fits the FOV for small VRX angles) making it impractical for evaluating the performance at large angles, or too big and only useful for relatively wide angles, defeating the purpose of the VRX scanner.

6.4 Conclusion

The study presented in this chapter analyzed the influence of the VRX angle on the severity of some types of CT artifacts. The results show that there is a direct relationship between the the VRX angle and the beam hardening artifacts produced by the interaction of highly absorbent objects, such as bones. It is suggested that this is due to the smaller size of the objects that fit in the FOV as the VRX angle decreases.

The relation between the VRX angle and the severity of streak artifacts produced by thin metallic objects is less clear. Depending on the detector device modeled, the error introduced by the streaks was found to either increase or decrease as the VRX angle increases. Furthermore, it was found that very small changes in the position of the phantom have an effect that may overshadow the effect being measured, making an experimental corroboration of these results diffucult at best.

The main conclusion that can be derived from the observations is that the VRX angle per se does not have a measurable deleterious effect on the generation and severity of the artifacts studied. In fact, images obtained at small VRX angles can be expected to have less beam hardening artifacts due to the smaller FOV.

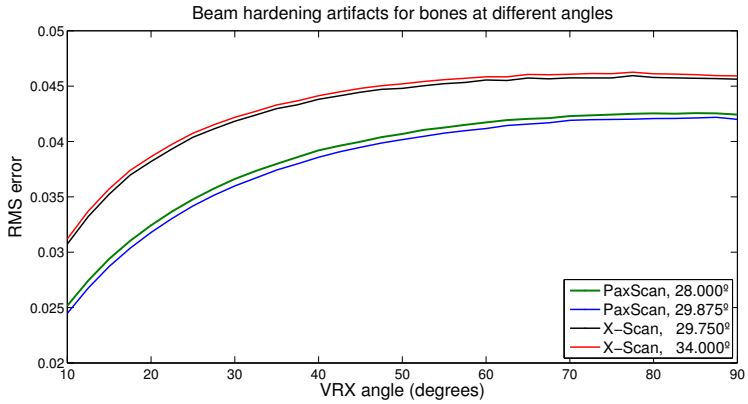


Figure 6.9: **Severity of the beam hardening artifacts in the bone phantoms as a function of the VRX angle for both detectors.**

Chapter 7

Completed and Future Work

In the process of pursuing the two main goals of this project, i.e., the development of computational tools for the study of VRX CT scanners and the application of said tools to help answer questions of interest to researchers of these devices, several smaller goals needed to be accomplished also. In this chapter we summarize these major and minor accomplishments, and we also present some additional projects in which we believe the computational tools will again prove to be valuable and practical.

7.1 Completed Work

In Chapter 3 we discussed the development of tools needed for the operation of multi-arm VRX scanners, both in the physical and in the virtual world. The code for these tools is included in Appendix A. These tools included calibration, reconstruction, and supporting algorithms:

- The reconstruction algorithm allows for an arbitrary number of arms in arbitrary positions. To accommodate these unusual geometries a significant departure from traditional reconstruction algorithms was necessary.
- The algorithm is capable of reconstructing images from sinograms acquired with both physical and virtual scanners.
- The algorithm is particularly well-suited for the reconstruction of target images, i.e., images in which the information for the central region is acquired at a much higher resolution than the external region.
- In most asymmetrical targeting VRX scanners the external region exits the field of view of the scanner during part of the acquisition process. This was taken into account by replacing the missing parts of the sinogram with data carefully selected from the parts that were acquired.
- Because of its architecture, the definition of new VRX scanners with particular characteristics to be used with this reconstruction algorithm (what we call different VRX configurations) becomes a trivial task.

- In fact, this algorithm can be easily adapted to other geometries going beyond the scope of VRX devices. The same can be said of all the other algorithms developed here because they share the same architecture.
- Current commercial CT scanners rely on specialized digital signal processing hardware that greatly accelerates the most time demanding tasks in the reconstruction process. In our algorithm, those bottlenecks have been identified and isolated so that an eventual integration with such specialized hardware would become almost trivial.
- Unfortunately, because of the costs involved we did not have access to such hardware at this time so our algorithms were optimized to take the best advantage possible of normal PC hardware.
- The sinograms acquired with VRX scanners, especially in the case of multi-arm devices, are extremely irregular demanding a step of interpolation prior to filtering and backprojection. We took advantage of the fact that such sinograms do have some structure to implement an interpolation routine that is at least an order of magnitude faster than those obtained by traditional methods.
- Among the supporting code, a successful ring correction image postprocessing algorithm was also developed.

In Chapter 4 we introduced the main virtual VRX scanners that we developed as a fundamental aid in the study and optimization of these devices, and included the code in Appendix B. The constraints in our computational resources constituted handicap but at the same time challenged us to focus on certain aspects of the simulators allowing us to achieve some interesting developments:

- We developed an interesting alternative to modeling complex single slice phantoms by representing complex objects through the combination of convex polygons and arbitrary elipses.
- The algorithms handle correctly the attenuation of polychromatic x-ray beams with realistic energy distributions.
- Due to its integration with Spect, our virtual scanners allow us to use phantoms composed of combinations of arbitrary materials, including materials that are hard or impossible to replicate in the lab such as pure water of different densities.
- Because they share the basic architecture of the reconstruction algorithm, our simulation algorithms can easily be adjusted to new VRX configurations and potentially to other CT geometries.
- Some of the physical phenomena included in the simulations such as noise, crosstalk, and others, can be controlled and even suppressed allowing for a better isolation of the features of interest in a particular virtual experiment.

- Although scattering is not presently included in the simulators, we proposed a method to approximate its effect in a way that can be integrated realistically dual current setup.
- We also determined optimal parameters that allow us to run simulations that reflect adequately the physical experiments being modeled in a shorter running time. These parameters included the number of energy bins in the polychromatic beam, the number of beams projected towards each detector cell, and the number of projection angles in a full rotation.

The tools we developed in the first part of the project were then used in two sample applications that demonstrate how these tools can be invaluable to VRX (and by extension CT) research. These two studies were presented in Chapters 5 and 6.

The first study compared three different configuration of otherwise equivalent target scanners, one symmetrical four-arm model and two asymmetrical two-arm configurations, in terms of spatial and contrast resolution:

- We used edge phantoms to determine the MTF and therefore the spatial resolution of the scanners in different regions of the field of view, and special contrast phantoms to determine the SNR_C and therefore the contrast resolution in the same regions.
- The study allowed us to conclude that the four-arm model and the Type B performed the best depending on the parameter measured and the region of the FOV, but that the Type A configuration tended to be inferior in general. This information is valuable when deciding what configurations are best to pursue with physical experiments in future research. In general the results with the Type B configuration were more consistently good, as the four-arm model performed very poorly in one region. Because of this, we conclude that the Type B two-arm scanner is the best of the three configurations.
- The study also allowed us to better understand the behavior of these parameters in the different regions of a targeting scanner. For example, it allowed us to realize that in the external region the spatial resolution in the tangential direction is significantly better than in the radial direction.

The final study (Chapter 6) examined the influence of the VRX effect on the severity of common artifacts present in the images generated by a single-arm VRX scanner:

- We used a phantom representing a thin metallic needle which in all common CT scanners produce streak artifacts, and phantoms of a pair of bones embedded in muscular tissue to analyze the beam hardening artifacts that will be produced between the bones.
- Because these are mathematical phantoms we know the exact composition of the phantom voxels represented by each of the pixels in the images. This allows

us to compare the actual images produced by the virtual scanner with the map of x-ray attenuation coefficients that a CT image would represent ideally. By analyzing how much the actual images deviate from the ideal ones we can quantify the severity of the artifacts.

- We found that the VRX angle does not have a measurable effect on the severity of these artifacts, other than effects that can be readily explained by the other experimental conditions such as the physical size of the phantom.

7.2 Future Work

Throughout this document we have emphasized repeatedly the importance and usefulness of the types of tools developed in this project. We have also given several examples of applications in which these tools have shown their value in research. But the potential applications are far more numerous and varied and what has been shown here. The tools could be used to assist in the design of future CT scanners, including identifying areas for which experimental research in design of the new system would be fruitless. For example, it would allow the developers to avoid expensive and time-consuming experimental studies of a system parameter if the simulator predicts that there will be no significant change in image performance.

There are also many improvements and modifications that could be done to the current tools widening even more their potential. For example, as the speed of personal computers continues to increase the possibility of incorporating the effects of scattering to the simulator becomes more feasible. Furthermore, these and other computationally intensive tasks could be made possible by the implementation of small computer clusters, or the use of specialized and yet relatively inexpensive hardware. In fact, in the last few years the graphics processor units (GPU) initially designed for gaming and now present in most personal computers have found their way into many other applications due to their ability to perform parallel calculations massively.

The tools could also be generalized to other CT architectures. In particular, it should be relatively straightforward to extend them to 3D. This would allow their use in the study of CBCT and helical scanners.

The virtual scanner could also be extended to simulate motion artifacts by allowing changes in the position of the phantom or parts of it as it rotates. This would allow the modeling of bipolar motion (i.e., when parts of the phantom alternate between two fixed positions), and uniform or oscillatory linear motion as well as circular motion and motion through complex paths.

Chapter 8

Conclusions

This dissertation presented a set of computational tools designed for the study and development of Variable Resolution X-Ray (VRX) CT scanners. The unique detector geometry of multi-arm VRX scanners presented us with special challenges that compelled us to develop our own tools that are fundamentally different from the ones traditionally used in this discipline.

These tools included a reconstruction algorithm that builds a parallel beam sinogram that can be easily reconstructed from the irregular sinograms generated by multi-arm VRX scanners, including two- and four-arm targeting scanners. Another important tool developed as part of this project was a comprehensive computer simulator of a single slice multi-arm VRX scanner. Due to the unique geometry of VRX scanners and to some practical limitations we had, the development of these tools presented some special challenges that required the implementation of some algorithms and methods that were either completely new or heavily modified variations of pre-existing ones. These achievements have been summarized in Section 7.1.

The development of the VRX simulator prompted us to study the effects of the different simulation parameters on the quality of the image generated. It was confirmed that even though the number of energy bins used to model a polychromatic x-ray beam will have the expected effect on energy dependent phenomena such as beam hardening, it has no measurable effect on the spatial resolution of the reconstructed image. On the other hand, the number of projection views in the sinogram does have a large influence on the resolution, an effect that becomes less noticeable for 800 views or more. We also concluded that, under the conditions we normally use for our simulations, the best combination of parameters (i.e., the one that reduces the simulation time by using smaller parameters without compromising the adequate reconstruction of the image quality), appears to be: 20 to 40 energy bins, 4 rays/cell, 800 views per full rotation.

The second part of this dissertation was devoted to the application of those computational tools to two studies that compared the performance of different types of VRX scanners under various conditions. One of the studies compared the spatial resolutions achieved by three different configurations of otherwise equivalent targeting VRX scanners: a four-arm scanner and two variations of asymmetric two-arm scanners which we called Type A and Type B. We found that the spatial resolution of targeting VRX scanners is different in the radial and azimuthal directions, especially outside the target region. Additionally, we found that the four-arm scanner has

a decent performance in the target area in terms of spatial and contrast resolution, but surprisingly it performs poorly in the external region. We also found that the Type A two-arm model had an inferior performance in all but one region of interest. The Type B two-arm model performed nearly as good as the four-arm one in the target region and in the external region in the tangential direction, and didn't perform nearly as bad in the radial direction. Due to its more consistent behavior we conclude that the Type B two-arm scanner is the best of the three configurations.

The other study analyzed the influence of the VRX angle on the severity of some types of CT artifacts. We found that the VRX angle per se does not have a measurable deleterious effect on the generation and severity of the artifacts studied. The results showed that there is a direct relationship between the the VRX angle and the beam hardening artifacts produced by the interaction of highly absorbent objects, such as bones. It was suggested that this is due to the smaller size of the objects that fit in the FOV as the VRX angle decreases. In fact, images obtained at small VRX angles can be expected to have less beam hardening artifacts due to the smaller FOV. No direct relationship was found between the VRX angle and the severity of streak artifacts produced by thin metallic objects, since depending on the detector device modeled, the error introduced by the streaks was found to either increase or decrease as the VRX angle increases.

List of References

[1] S. Achenbach, D. Ropers, A. Kuettner, T. Flohr, B. Ohnesorge, H. Bruder, H. Theessen, M. Karakaya, W. G. Daniel, W. Bautz, W. A. Kalender, and K. Anders, "Contrast-enhanced coronary artery visualization by dual-source computed tomography–initial experience," *Eur J Radiol*, vol. 57, no. 3, pp. 331–335, 2006.

[2] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand *et al.*, "Geant4 - a simulation toolkit," *Nuclear Instruments and Methods in Physics Research A*, vol. 506, no. 3, pp. 250–303, 2003.

[3] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Dubois, and M. Asai, "Geant4 developments and applications," *IEEE Transactions on Nuclear Science*, vol. 53, no. 1, pp. 270–278, 2006.

[4] A. Badal, I. Kyprianou, A. Badano, J. Sempau, and K. J. Myers, "Monte Carlo package for simulating radiographic images of realistic anthropomorphic phantoms described by triangle meshes," in *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 6510, no. 1, 2007.

[5] A. Badal and J. Sempau, "A package of Linux scripts for the parallelization of Monte Carlo simulations," *Computer Physics Communications*, vol. 175, no. 6, pp. 440–450, 2006.

[6] J. F. Barrett and N. Keat, "Artifacts in CT: recognition and avoidance," *Radiographics*, vol. 24, no. 6, pp. 1679–1691, 2004.

[7] S. J. Boyce and E. Samei, "Imaging properties of digital magnification radiography," *Med Phys*, vol. 33, no. 4, pp. 984–996, 2006.

[8] J. T. Bushberg, J. A. Seibert, E. M. L. Jr., and J. M. Boone, *The Essential Physics of Medical Imaging*, 2nd ed. Lippincott Williams & Wilkins, Philadelphia, PA, 2001.

[9] A. P. Colijn, W. Zbijewski, A. Sasov, and F. J. Beekman, "Experimental validation of a rapid Monte Carlo based micro-CT simulator," *Phys Med Biol*, vol. 49, no. 18, pp. 4321–4333, 2004.

[10] B. Dahi, G. S. Keyes, D. A. Rendon, and F. A. DiBianca, "Performance analysis of a CsI-based flat panel detector in a cone beam variable resolution x-ray system," in *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 6510, no. 3, 2007.

[11] B. Dahi, G. S. Keyes, D. A. Rendon, and F. A. Dibianca, "Analysis of axial spatial resolution in a variable resolution x-ray cone beam CT (VRX-CBCT) system," in *Proceedings of SPIE*, vol. 6913, 2008.

[12] F. R. de Graaf, J. D. Schuijf, V. Delgado, J. E. van Velzen, L. J. Kroft, A. de Roos, J. W. Jukema, E. E. van der Wall, and J. J. Bax, "Clinical application of CT coronary angiography: state of the art," *Heart Lung Circ*, vol. 19, no. 3, pp. 107–116, Mar 2010.

[13] F. A. DiBianca, D. Gulabani, L. M. Jordan, S. Vangala, D. Rendon, J. S. Laughter, R. Melnyk, M. W. Gaber, and G. S. Keyes, "Four-arm variable-resolution x-ray detector for CT target imaging," in *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 5745, no. I, 2005, pp. 332–339.

[14] F. A. DiBianca, V. Gupta, and H. D. Zeman, "A variable resolution x-ray detector for computed tomography: I. Theoretical basis and experimental verification," *Med Phys*, vol. 27, no. 8, pp. 1865–1874, 2000.

[15] F. A. DiBianca, R. Melnyk, C. Duckworth, S. Russ, L. M. Jordan, and J. S. Laughter, "Comparison of VRX CT scanner geometries," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4320, 2001, pp. 627–635.

[16] F. A. DiBianca, P. Zou, L. M. Jordan, J. S. Laughter, H. D. Zeman, and J. Sebes, "A variable resolution x-ray detector for computed tomography: II. Imaging theory and performance," *Med Phys*, vol. 27, no. 8, pp. 1875–1880, 2000.

[17] F. A. DiBianca, V. Gupta, P. Zou, L. M. Jordan, J. S. Laughter, H. D. Zeman, and J. I. Sebes, "Ultrahigh-resolution CT and DR scanner," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3659, no. I, pp. 56–64, 1999.

[18] F. A. DiBianca, R. Melnyk, A. Sambari, L. M. Jordan, J. S. Laughter, and P. Zou, "Solid-state VRX CT detector," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3977, 2000, pp. 205–210.

[19] J. Dobbins, *Handbook of Medical Imaging: Physics and Psychophysics*. SPIE Publications, Bellingham, WA, 2000, vol. 1, ch. Image quality metrics for digital systems.

[20] R. T. Droege and R. L. Morin, "A practical method to measure the MTF of CT scanners," *Med Phys*, vol. 9, no. 5, pp. 758–760, 1982.

[21] I. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," *Journal of the Optical Society of America A: Optics and Image Science, and Vision*, vol. 1, no. 6, pp. 612–619, 1984.

[22] N. L. Ford, M. M. Thornton, and D. W. Holdsworth, "Fundamental image quality limits for microcomputed tomography in small animals," *Med Phys*, vol. 30, no. 11, pp. 2869–2877, 2003.

[23] M. W. Gaber, C. M. Wilson, C. D. Duntsch, H. Shukla, J. A. Zawaski, L. M. Jordan, D. A. Rendon, S. Vangala, G. S. Keyes, and F. A. DiBianca, "Volumetric analysis of tumors in rodents using the variable resolution x-ray (VRX) CT-scanner," in *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 5746, no. I, 2005, pp. 9–15.

[24] M. L. Giger and K. Doi, "Investigation of basic imaging properties in digital radiography. I. Modulation transfer function," *Med Phys*, vol. 11, no. 3, pp. 287–295, 1984.

[25] D. Goodenough, *Handbook of Medical Imaging: Physics and Psychophysics*. SPIE Publications, Bellingham, WA, 2000, vol. 1, ch. Tomographic imaging, pp. 511–558.

[26] D. Gopalan, V. Raj, and E. T. D. Hoey, "Cardiac CT: non-coronary applications," *Postgrad Med J*, vol. 86, no. 1013, pp. 165–173, Mar 2010.

[27] J. E. Gray and M. Trefler, "Phase effects in diagnostic radiological images," *Med Phys*, vol. 3, no. 4, pp. 195–203, 1976.

[28] W. Haddad, I. McNulty, J. Trebes, E. Anderson, R. Levesque, and L. Yang, "Ultrahigh-resolution x-ray tomography," *Science*, vol. 266, no. 5188, pp. 1213–1215, 1994.

[29] J. A. Halbleib, R. P. Kensek, G. D. Valdez, S. M. Seltzer, and M. J. Berger, "ITS: The integrated TIGER series of electron/photon transport codes–version 3.0," *IEEE Transactions on Nuclear Science*, vol. 39, no. 4, pp. 1025–1030, 1992.

[30] K. Hashimoto, Y. Arai, K. Iwai, M. Araki, S. Kawashima, and M. Terakado, "A comparison of a new limited cone beam computed tomography machine for dental use with a multidetector row helical CT machine," *Oral Surg Oral Med Oral Pathol Oral Radiol Endod*, vol. 95, no. 3, pp. 371–377, Mar 2003.

[31] D. W. Holdsworth and M. M. Thornton, "Micro-CT in small animal and specimen imaging," *Trends in Biotechnology*, vol. 20, no. 8, pp. S34–S39, 2002.

[32] J. Hsieh, *Computed Tomography. Principles, Design, Artifacts, and Recent Advances*. SPIE Publications, Bellingham, WA, 2003.

[33] T. R. C. Johnson, B. Krauß, M. Sedlmair, M. Grasruck, H. Bruder, D. Morhard, C. Fink, S. Weckbach, M. Lenhard, B. Schmidt, T. Flohr, M. F. Reiser, and C. R. Becker, "Material differentiation by dual energy CT: initial experience," *Eur Radiol*, vol. 17, no. 6, pp. 1510–1517, Jun 2007.

[34] L. M. Jordan, F. A. DiBianca, R. Melnyk, A. Choudhary, H. Shukla, J. Laughter, and M. W. Gaber, "Determination of calibration parameters of a VRX CT system using an 'amoeba' algorithm," *Journal of X-Ray Science and Technology*, vol. 12, no. 4, pp. 281–293, 2004.

[35] L. M. Jordan, F. A. DiBianca, Z. Ping, J. Laughter, and H. Zeman, "Processing of CT sinograms acquired using a VRX detector," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3977, 2000, pp. 570–579.

[36] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. New York, NY, USA: IEEE Press, 1988.

[37] W. A. Kalender, "X-ray computed tomography," *Phys Med Biol*, vol. 51, no. 13, pp. R29–43, 2006.

[38] W. M. Kohrt, "Preliminary evidence that DEXA provides an accurate assessment of body composition," *J Appl Physiol*, vol. 84, no. 1, pp. 372–377, 1998.

[39] S. C. Lee, H. K. Kim, I. K. Chun, M. H. Cho, S. Y. Lee, and M. H. Cho, "A flat-panel detector based micro-CT system: performance evaluation for small-animal imaging," *Phys Med Biol*, vol. 48, no. 24, pp. 4173–4185, 2003.

[40] R. Melnyk and F. A. DiBianca, "Monte Carlo study of x-ray cross-talk in a variable resolution x-ray detector," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5030 II, 2003, pp. 694–701.

[41] R. Melnyk and F. A. DiBianca, "Modeling and measurement of the detector presampling MTF of a variable resolution x-ray CT scanner," *Med Phys*, vol. 34, no. 3, pp. 1062–1075, 2007.

[42] C. E. Metz and K. Doi, "Transfer function analysis of radiographic imaging systems," *Phys Med Biol*, vol. 24, no. 6, pp. 1079–1106, 1979.

[43] J. Morishita, K. Doi, R. Bollen, P. C. Bunch, D. Hoeschen, G. Sirand-rey, and Y. Sukenobu, "Comparison of two methods for accurate measurement of modulation transfer functions of screen-film systems," *Med Phys*, vol. 22, no. 2, pp. 193–200, 1995.

[44] P. Mozzo, C. Procacci, A. Tacconi, P. T. Martini, and I. A. Andreis, "A new volumetric CT machine for dental imaging based on the cone-beam technique: preliminary results," *Eur Radiol*, vol. 8, no. 9, pp. 1558–1564, 1998.

[45] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[46] E. L. Nickoloff and R. Riley, "A simplified approach for modulation transfer function determinations in computed tomography," *Med Phys*, vol. 12, no. 4, pp. 437–442, 1985.

[47] R. Ning, X. Tang, and D. Conover, "X-ray scatter correction algorithm for cone beam CT imaging," *Med Phys*, vol. 31, no. 5, pp. 1195–1202, 2004.

[48] J. N. O'Dwyer and J. R. Tickner, "Modelling diffractive X-ray scattering using the EGS Monte Carlo code," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 580, no. 1 SPEC. ISS., pp. 127–129, 2007.

[49] M. Ohkubo, S. Wada, T. Matsumoto, and K. Nishizawa, "An effective method to verify line and point spread functions measured in computed tomography," *Med Phys*, vol. 33, no. 8, pp. 2757–2764, 2006.

[50] M. J. Paulus, S. S. Gleason, S. J. Kennel, P. R. Hunsicker, and D. K. Johnson, "High resolution x-ray computed tomography: an emerging tool for small animal cancer research," *Neoplasia*, vol. 2, no. 1-2, pp. 62–70, 2000.

[51] D. A. Rendon, F. A. DiBianca, and G. S. Keyes, "Comparison of multi-arm VRX CT scanners through computer models," in *Progress in Biomedical Optics and Imaging - Proceedings of SPIE*, vol. 6510, no. 2, 2007.

[52] D. A. Rendon, F. A. DiBianca, and G. S. Keyes, "Reconstruction artifacts in VRX CT scanner images," in *Proceedings of SPIE*, vol. 6913, 2008.

[53] E. Ritman, "Molecular imaging in small animals–roles for micro-CT," *J Cell Biochem Suppl*, vol. 39, pp. 116–124, 2002.

[54] E. L. Ritman, "Micro-computed tomography-current status and developments," *Annu Rev Biomed Eng*, vol. 6, pp. 185–208, 2004.

[55] S. Robinson, A. Suomalainen, and M. Kortesniemi, "μ-CT," *Eur J Radiol*, vol. 56, no. 2, pp. 185–191, 2005.

[56] K. Rossmann, "Point spread-function, line spread-function, and modulation transfer function. tools for the study of imaging systems," *Radiology*, vol. 93, no. 2, pp. 257–272, 1969.

[57] E. Samei, M. J. Flynn, and D. A. Reimann, "A method for measuring the presampled MTF of digital radiographic systems using an edge test device," *Med Phys*, vol. 25, no. 1, pp. 102–113, 1998.

[58] E. Samei, N. T. Ranger, J. T. Dobbins, and Y. Chen, "Intercomparison of methods for image quality characterization. I. Modulation transfer function," *Med Phys*, vol. 33, no. 5, pp. 1454–1465, 2006.

[59] A. Sasov, "Comparison of fan-beam, cone-beam and spiral scan reconstruction in X-ray micro-CT," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4503, 2001, pp. 124–131.

[60] A. Sasov, "Desktop x-ray micro-CT instruments," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4503, pp. 282–290, 2001.

[61] A. Sasov and D. Dewaele, "High-resolution in-vivo micro-CT scanner for small animals," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4503, pp. 256–264, 2001.

[62] J. Sempau, E. Acosta, J. Baró, J. M. Fernández-Varea, and F. Salvat, "An algorithm for Monte Carlo simulation of coupled electron-photon transport," *Nuclear Instruments and Methods in Physics Research, Section B: Beam Interactions with Materials and Atoms*, vol. 132, no. 3, pp. 377–390, 1997.

[63] L. A. Shepp and B. F. Logan Jr., "Fourier reconstruction of a head section," *IEEE Transactions on Nuclear Science*, vol. NS-21, no. 3, pp. 21–43, 1974.

[64] J. H. Siewerdsen, D. J. Moseley, B. Bakhtiar, S. Richard, and D. A. Jaffray, "The influence of antiscatter grids on soft-tissue detectability in cone-beam computed tomography with flat-panel detectors," *Med Phys*, vol. 31, no. 12, pp. 3506–3520, 2004.

[65] P. Stolzmann, R. Goetti, S. Baumueller, A. Plass, V. Falk, H. Scheffel, G. Feuchtner, B. Marincek, H. Alkadhi, and S. Leschka, "Prospective and retrospective ECG-gating for CT coronary angiography perform similarly accurate at low heart rates," *Eur J Radiol*, Jan 2010.

[66] G. Wang, C. R. Crawford, and W. A. Kalender, "Multirow detector and cone-beam spiral/helical CT," *IEEE Trans Med Imaging*, vol. 19, no. 9, pp. 817–821, Sep 2000.

[67] G. Wang, S. Zhao, H. Yu, C. A. Miller, P. J. Abbas, B. J. Gantz, S. W. Lee, and J. T. Rubinstein, "Design, analysis and simulation for development of the first clinical micro-CT scanner," *Acad Radiol*, vol. 12, no. 4, pp. 511–525, 2005.

[68] A. Webb, *Introduction to Biomedical Imaging*. Wiley-IEEE Press, Piscataway, NJ, 2003.

[69] R. Weissleder and U. Mahmood, "Molecular imaging," *Radiology*, vol. 219, no. 2, pp. 316–333, 2001.

[70] R. Weissleder and M. J. Pittet, "Imaging in the era of molecular oncology," *Nature*, vol. 452, no. 7187, pp. 580–589, 2008.

[71] K. Wiesent, K. Barth, N. Navab, P. Durlak, T. Brunner, O. Schuetz, and W. Seissler, "Enhanced 3-D-reconstruction algorithm for C-arm systems suitable for interventional procedures," *IEEE Trans Med Imaging*, vol. 19, no. 5, pp. 391–403, May 2000.

[72] L. Xu and Z. Zhang, "Coronary CT angiography with low radiation dose," *Int J Cardiovasc Imaging*, Jan 2010.

Appendix A

Multi-arm VRX Reconstruction Code

The reconstruction of tomographic images from sinograms acquired with multi-arm VRX scanners described in Section 3.1 is performed mainly by the MATLAB program `reconstr_VRX_fan_angle.m`, which will be listed in Section A.1. The reconstruction program depends on auxiliary programs that determine the configuration (`get_VRX_configuration.m`, listed in Section A.2) and recover the calibration parameters (`get_VRX_calibration.m`, Section A.3). The actual back-projection is performed by a C-Mex function (that is, a function written in C and compiled into a library that can be accessed from MATLAB like a normal function) called `f_iradon.c`, which is called by `fast_iradon.m`, a MATLAB function that also does the sinogram filtering. These functions are listed in Section A.4. As the name implies, this is an highly optimized function that reduces the main bottleneck of CT reconstruction. An additional post-processing step is performed for images acquired with some devices which tend to produce concentric rings around the center of the image, a problem common in third generation CT scanners that was described in Section 2.1.1. This ring correction algorithm (`ring_corrector.m`) was described in Section 3.1.4 and is listed in Section A.5. The calibration of multi-arm VRX scanners (Section 3.11) is performed by the program `vrx_calibration.m` which is listed in Section A.6.

A.1 Reconstruction Program

```
function [reconsImage,pixelSize] = reconstr_VRX_fan_angle (varargin)
% Reconstructs a tomographic slice aquired with a fan-angle beam on a
% VRX scanner.
%
% [reconsImage,pixelSize] = reconstr_VRX_fan_angle (inputSinog, ...
%       numDesiredBeams, sizeRecons, calibrationFile, vrxConfig);
%
% Inputs:
%   inputSinog      -   a matrix with the sinogram or a file name
%   numDesiredBeams -   number of pixels covering the whole FOV
%   sizeRecons      -   number of pixels in the reconstructed image
%   calibrationFile -   file with calibration info
%   vrxConfig       -   kind of VRX scanner ('2 arms', '4 arms',
%                       'x-scan', etc)
```

```
%
% Outputs:
%   reconsImage     -   reconstructed image
%   pixelSize       -   size in cm of a pixel in the reconstructed
%                       image
%


[inputSinog, numDesiredBeams, sizeRecons, calibrationFile, ...
    vrxConfig] ...
    = parse_inputs(varargin{:});

[numArms, cellPositions, cellWidth] ...
    = get_VRX_configuration (vrxConfig);
[numViews, distFScenter, psi, distOnArm, distFromTube] ...
    = get_VRX_calibration (calibrationFile, numArms);

totalViews = size(inputSinog,1);
difViews = (totalViews - numViews)/2;

[cellAngles, cell2FS, cellAngleCover, totalAngle] ...
    = fan_angles (cellPositions, cellWidth, psi, distOnArm, ...
        distFromTube);
cellAngles = cellAngles(:)';
cell2FS = cell2FS(:);
anglePortions = cellAngleCover(:)' / totalAngle;

% Do any preprocessing needed on the sinogram.
inputSinog = log(inputSinog ./ repmat(anglePortions, totalViews, 1));
inputSinog = -inputSinog + max(inputSinog(:));


tic
parallelSinog = fan2parallel (inputSinog, numViews, cellAngles, ...
    distFScenter, numDesiredBeams);

if totalViews > numViews,
    parallelSinog = parallelSinog ...
        (floor(difViews)+1:end-ceil(difViews), :);
else
    numViews = totalViews;
end
toc

tic
%ag = linspace(0, 360, size(parallelSinog,1)+1);
%reconsImage = iradon(parallelSinog', ag(1:end-1), 'nearest',  ...
%       'Shepp-Logan', 1, sizeRecons);
reconsImage = fast_iradon(parallelSinog', sizeRecons);
toc

global makePlots
if isempty (makePlots)
    makePlots = false;
```

```
end
if makePlots
    figure
    imagesc(reconsImage); colormap gray; axis image
end

%reconsImage  = (reconsImage - min(reconsImage(:))) / ...
%       (max(reconsImage(:)) - min(reconsImage(:)));
%imwrite(reconsImage, ['wrist 23.png']);
%imwrite(reconsImage, ['wrist 22 ' num2str(ii) '.png']);




%% ---
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Internal functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [inputSinog, numDesiredBeams, sizeRecons, ...
    calibrationFile, vrxConfig] = parse_inputs (varargin)
%  Parse the input arguments and retun things
%
%  Inputs:   varargin -   Cell array containing all of the actual
%                         inputs
%
%  Outputs:  p        -   Projection data

if nargin < 1,
    error('Provide at least the input sinogram.');
end

if isnumeric(varargin{1}),
    % The sinogram was already provided as a matrix
    inputSinog = varargin{1};
elseif strcmp(lower(varargin{1}(end-3:end)), '.mat')
    % The sinogram is saved in a .MAT file in a variable called sino
    load(varargin{1});
    inputSinog = double(sino);
else  strcmp(lower(varargin{1}(end-3:end)), '.dat')
    % The sinogram is in a binary file
    %(old VRX data preprocessed in IDL)
    fid = fopen(varargin{1}, 'r');
    % Roman's data modif
    % original lines:
    inputSinog = fscanf(fid, '%f', [576,inf])';
    fclose(fid);
    inputSinog = flipud(inputSinog);

    % new lines (for Roman's special data):
%     inputSinog = fread(fid, [576,inf], 'float32', 0, 'l')';
%     fclose(fid);
%     %inputSinog = flipud(inputSinog);
```

```
end


if nargin < 2 || isempty(varargin{2}),
    numDesiredBeams = 100;
else
    numDesiredBeams = varargin{2};
end

if nargin < 3 || isempty(varargin{3}),
    sizeRecons = 100;
else
    sizeRecons = varargin{3};
end

if nargin < 4 || isempty(varargin{4}),
    calibrationFile = './caldelxdata.dat';
else
    calibrationFile = varargin{4};
end

if nargin < 5 || isempty(varargin{5}),
    vrxConfig = '4 arms';
else
    vrxConfig = varargin{5};
end



%% ---
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [cellAngles, cell2FS, cellAngleCover, totalAngle] ...
    = fan_angles ...
                (cellPositions, cellWidth, psi, distOnArm, ...
                   distFromTube)
% Finds the fan angle at which each cell of a multi-arm single slice
% VRX scanner is located (relative to the focal spot-center of
% rotation line).
%
% Angles in radians, negative for left arm, positive for right arm.
%
% cellPositions: distance in cm to each cell from reference point of
%   the arm. (columns represent the different arms).
% psi: angle formed by fs-cor line and each arm (in radians).
% distOnArm: distance to intersection of fs-cor line and each arm,
%   from reference point.
% distFromTube: distance from the intersections to the focal spot.
%
%

[numCells, numArms] = size(cellPositions);

% Replicate arrays so that corresponding cells in cellPositions get
```

```
% the info for their arm.
psi = repmat (psi, numCells, 1);
distOnArm = repmat (distOnArm, numCells, 1);
distFromTube = repmat (distFromTube, numCells, 1);

% Find the distances to each cell from the corresponding intersection
% point.
cellDistances = cellPositions - distOnArm;

% Find the angles (in radians) using sine/cosine laws and some ...
   algebra
cellX = distFromTube - cellDistances .* cos(psi);
cellY = cellDistances .* sin(psi);
cell2FS = sqrt(cellX.^2 + cellY.^2);

cellAngles = atan(cellY ./ cellX);


% Repeat for the left and right boundaries of the cells to obtain the
% arcs subtended.
cellDistLeft = cellDistances - cellWidth/2;
cellXL = distFromTube - cellDistLeft .* cos(psi);
cellYL = cellDistLeft .* sin(psi);
cellAnglesLeft = atan(cellYL ./ cellXL);


cellDistRight = cellDistances + cellWidth/2;
cellXR = distFromTube - cellDistRight .* cos(psi);
cellYR = cellDistRight .* sin(psi);
cellAnglesRight = atan(cellYR ./ cellXR);

cellAngleCover = cellAnglesRight - cellAnglesLeft;
totalAngle = cellAnglesRight(end) - cellAnglesLeft(1);



%% --- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function parallelSinog = fan2parallel (fanSinog, numViews, ...
    fanAngles, distFScenter, numDesiredBeams)

totalViews = size(fanSinog,1);

[beta, phi] = meshgrid (fanAngles, (1:totalViews)/numViews*2*pi);

theta = beta + phi;
l = distFScenter * sin(fanAngles);

[l, theta, completeSinog] = add_virtual_arms (l, theta, fanSinog);

halfSpan = min(abs([min(l(:)), max(l(:))]));
pixelSize = 2*halfSpan / numDesiredBeams
desiredBeams = linspace(-halfSpan, halfSpan, numDesiredBeams);
desiredThetas = (1:totalViews)'/numViews*2*pi;
```

```
parallelSinog = cust_interp (l, theta, completeSinog, ...
    desiredBeams, desiredThetas);


%% --- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [l, theta, completeSinog] = ...
    add_virtual_arms (l, theta, fanSinog)

lRow = l(1,:);
halfViews = round(size(fanSinog,1)/2) + 6;

if abs(lRow(1)) < abs(lRow(end)),
    % Add cells from the right as virtual cells to the left
    virtualArms = lRow(abs(lRow(1)) < abs(lRow));
    virtualSinog = fliplr( ...
        fanSinog(:, end-length(virtualArms)+1:end) );
    virtualSinog = [virtualSinog(halfViews+1:end,:); ...
        virtualSinog(1:halfViews,:)];

    virtualL = -fliplr(l(:, end-length(virtualArms)+1:end));

    virtualTheta = fliplr(theta(:, end-length(virtualArms)+1:end));
    vTheta1 = virtualTheta(halfViews+1:end,:) - pi;
    vTheta2 = virtualTheta(1:halfViews,:) + pi;
    virtualTheta = [vTheta1; vTheta2];

    completeSinog = [virtualSinog fanSinog];
    l = [virtualL l];
    theta = [virtualTheta theta];
else
    % Add cells from the left as virtual cells to the right
    virtualArms = lRow(abs(lRow(end)) < abs(lRow));
    virtualSinog = fliplr(fanSinog(:, 1:length(virtualArms)));
    virtualSinog = [virtualSinog(halfViews+1:end,:); ...
        virtualSinog(1:halfViews,:)];

    virtualL = -fliplr(l(:, 1:length(virtualArms)));

    virtualTheta = fliplr(theta(:, 1:length(virtualArms)));
    vTheta1 = virtualTheta(halfViews+1:end,:) - pi;
    vTheta2 = virtualTheta(1:halfViews,:) + pi;
    virtualTheta = [vTheta1; vTheta2];

    completeSinog = [fanSinog virtualSinog];
    l = [l virtualL];
    theta = [theta virtualTheta];
end


%% --- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function parSin = cust_interp (l, theta, fanSinog, desiredBeams, ...
    desiredThetas)
% Obtain a (normally) regular parallel beam sinogram by interpolating
% on an irregular fan beam sinogram (e.g., from a VRX scanner).
%
%   l:      For each fan beam (line from focal spot to one detector),
%           the distance to it from the center of rotation.
%           (Independent of the view angle).
%   theta:  For each fan beam, theta = beta + phi, that is, the view
%           angle (phi) would follow the same path thorugh the
%           phantom.
%   fanSinog: Fan beam sinogram. Each row is a view, each column a
%           fan beam.
%   desiredBeams: Distances from central beam (FS-center of rot axis)
%           to each of the desired parallel beams. Normally (but not
%           necessarily), regularly distributed.
%   desiredThetas: Vector with view angles in the desired sinogram.
%           Normally regularly distributed.

% Get the original and desired number of beams and views.
nl = length(l);
nThetas = size(theta, 1);
nDesBeams = length (desiredBeams);
nDesThetas = length (desiredThetas);

% For each desired parallel beam, we need to find the two detectors
% within whose parallelized beams it would lie. This is independent
% of the view angle (phi or theta).
%
% We do this easily by building matrices to compare all the original
% beam distances (l) with all the desired beam distances
% (desiredBeams), with each column representing a desired beam.
lMat = repmat (l', 1, nDesBeams);
desBeamsMat = repmat(desiredBeams, nl, 1);

% For each desired beam, get the index of the l that is closest <= to
% it. The next index should be the closest > it.
l0 = sum(lMat <= desBeamsMat);
l1 = l0 + 1;
% Make sure you don't leave the valid index range (1:number of
% detectors).
l0 (l0 < 1) = 1;
l1 (l1 > nl) = nl;

% For each desired view angle, we need the two thetas within which it
% would lie. Since theta depends both on phi (original view angle)
% and the original fan angle beta (or equivalently the detector
% index), we will need to store them in two matrices (theta0, theta1)
% with desired view angles in each row and detector index in each
% column. We build theta 0 one column at a time.
theta0 = zeros(nDesThetas, nl);
desThetaMat = repmat(desiredThetas', nThetas, 1);
```

```
for ll = 1:nl,
    % Build theta0 for the detector with index ll.
    dt = 1;
    for tt = 1:nThetas,
        % For each theta in the sinogram, identify the desired thetas
        % that would have this one as their theta0.
        while desiredThetas(dt) < theta(tt, ll) && (dt < nDesThetas),
            theta0(dt, ll) = tt - 1;
            dt = dt+1;
        end
    end
    for dt2 = dt:nDesThetas,
        % All the desired thetas over the maximum one get that max
        % one as both theta0 and theta1.
        theta0(dt2, ll) = nThetas;
    end
end
% We have identified the theta0's, get the corresponding theta1's.
theta1 = theta0 + 1;
% Make sure you don't leave the valid index range (1:original number
% of thetas).
theta0 (theta0 < 1) = 1;
theta1 (theta1 > nThetas) = nThetas;

parSin = zeros(nDesThetas, nDesBeams);

warning off MATLAB:divideByZero
for bb = 1:nDesBeams,
    % For each desired beam:
    % Get the original sinogram values of each corner, for all the
    % thetas.
    s00 = fanSinog(theta0(:,l0(bb)), l0(bb));
    s10 = fanSinog(theta1(:,l0(bb)), l0(bb));
    s01 = fanSinog(theta0(:,l1(bb)), l1(bb));
    s11 = fanSinog(theta1(:,l1(bb)), l1(bb));

    % Interpolate on the corners corresponding to the previous
    % detector.
    d = desiredThetas(:) - theta(theta0(:,l0(bb)), l0(bb));
    n = theta(theta1(:,l0(bb)), l0(bb)) ...
        - theta(theta0(:,l0(bb)), l0(bb));
    d = d ./ n;
    d(n == 0) = 0;
    a0 = s00.*(1-d) + s10.*d;

    % Interpolate on the corners corresponding to the next detector.
    d = desiredThetas(:) - theta(theta0(:,l1(bb)), l1(bb));
    n = theta(theta1(:,l1(bb)), l1(bb)) ...
        - theta(theta0(:,l1(bb)), l1(bb));
    d = d ./ n;
    d(n == 0) = 0;
    a1 = s01.*(1-d) + s11.*d;

    % Interpolate on the previously interpolated values, along the
```

```
    % other direction. Place the results in the new sinogram.
    d2 = desiredBeams(bb) - l(l0(bb));
    n2 = l(l1(bb)) - l(l0(bb));
    if n2 > 0,
        d2 = d2 / n2;
    else
        d2 = 0;
    end
    parSin(:, bb) = a0*(1-d2) + a1*d2;
end
warning on MATLAB:divideByZero

% The following double for loop is equivalent to the previous
% (partially vetorized) simple for loop. The simple loop is much
% faster, but the double version was left here because it's easier
% to understand.
%
% for bb = 1:nDesBeams,
%     for dt = 1:nDesThetas,
%         % For each theta in each desired beam:
%         % Get the original sinogram values of each corner.
%         s00 = fanSinog(theta0(dt,l0(bb)), l0(bb));
%         s10 = fanSinog(theta1(dt,l0(bb)), l0(bb));
%         s01 = fanSinog(theta0(dt,l1(bb)), l1(bb));
%         s11 = fanSinog(theta1(dt,l1(bb)), l1(bb));
%
%         % Interpolate on the corners corresponding to the previous
%         % detector.
%         d = desiredThetas(dt) - theta(theta0(dt,l0(bb)), l0(bb));
%         n = theta(theta1(dt,l0(bb)), l0(bb)) ...
%               - theta(theta0(dt,l0(bb)), l0(bb));
%         if n > 0,
%             d = d / n;
%         else
%             d = 0;
%         end
%         a0 = s00*(1-d) + s10*d;
%
%         % Interpolate on the corners corresponding to the next
%         % detector.
%         d = desiredThetas(dt) - theta(theta0(dt,l1(bb)), l1(bb));
%         n = theta(theta1(dt,l1(bb)), l1(bb)) ...
%               - theta(theta0(dt,l1(bb)), l1(bb));
%         if n > 0,
%             d = d / n;
%         else
%             d = 0;
%         end
%         a1 = s01*(1-d) + s11*d;
%
%         % Interpolate on the previously interpolated values, along
%         % the other direction.
%         % Place the result in the new sinogram.
%         d = desiredBeams(bb) - l(l0(bb));
```

```
%         n = l(l1(bb)) - l(l0(bb));
%         if n > 0,
%             d = d / n;
%         else
%             d = 0;
%         end
%         parSin(dt, bb) = a0*(1-d) + a1*d;
%     end
% end
```

A.2 Description of the System Cofiguration

```
function [numArms, cellPositions, interCellDist, numCells] ...
        = get_VRX_configuration (vrxConfig)
%
% Return information on the arms of the VRX scanner and the position
% of the cells within the arms.

if nargin < 1 || isempty(vrxConfig),
    vrxConfig = '4 arms';
end

switch lower(vrxConfig)
    case '4 arms'
        numArms = 4;
        numCells = 144;                 % Num of cells per arm.
        interCellDist = .099;           % In cm.
        cellsPerMod = 24;               % Cells per module.
        interModDist = 2.402 - ...
            cellsPerMod*interCellDist;  % Extra gap between modules.
    case '2 arms'
        numArms = 2;
        numCells = 288;                 % Num of cells per arm.
        interCellDist = .099;           % In cm.
        cellsPerMod = 24;               % Cells per module.
        interModDist = 2.402 - ...
            cellsPerMod*interCellDist;  % Extra gap between modules.
    case 'x-scan'
        numArms = 1;
        numCells = 768;                 % Num of cells per arm.
        interCellDist = 32.72/numCells; % In cm.
        cellsPerMod = 768;              % Cells per module.
        interModDist = 0;
    case 'fpd slice 1024'
        numArms = 1;
        numCells = 1020;                % Num of cells per arm.
        interCellDist = 0.0194;         % In cm.
        cellsPerMod = 1020;             % Cells per module.
        interModDist = 0;
    case 'fpd slice 512'
```

```
        numArms = 1;
        numCells = 508;                 % Num of cells per arm.
        interCellDist = 0.0194*2;       % In cm.
        cellsPerMod = 508;              % Cells per module.
        interModDist = 0;
end

% Build the cell positions relative to the first cell for a single
% arm.
cellPositions = 0:(numCells-1);
cellPositions = cellPositions * interCellDist + ...
    floor(cellPositions/cellsPerMod) * interModDist;
% cellPositions has cell indexes in the rows, arms in the columns.
cellPositions = repmat(cellPositions', 1, numArms);
```

A.3 Description of the System Calibration

```
function [numViews, distFScenter, psi, distOnArm, distFromTube] ...
    = get_VRX_calibration (calibSource, numArms);
% Return information on the exact position of all the elements of the
% system (arms, center of rotation, x-ray tube)

if nargin < 1 || isempty(calibSource),
    calibSource = '../Jordan/Four_arms_VRX/caldelxdata.dat';
end

if nargin < 2,
    numArms = 1;
end

if isstruct(calibSource)
    % The calibration was given explicitly in a structure.
    distFromTube = calibSource.distFromCOR;
    distOnArm    = calibSource.distOnArm;
    psi          = calibSource.psi;
    distFScenter = calibSource.distFScenter;
    numViews     = calibSource.cycleViews;
elseif strcmp(calibSource(end-3:end), '.mat')
    % New style of calibration file, everything is in a structure in
    % a MAT file.
    load (calibSource);
    distFromTube = calibValues.distFromCOR;
    distOnArm    = calibValues.distOnArm;
    psi          = calibValues.psi;
    distFScenter = calibValues.distFScenter;
    numViews     = calibValues.cycleViews;
else
    % Assume old style: all parameters are in a text file
    fid = fopen(calibSource, 'r');
    distFromTube = fscanf(fid, '%f', [1,numArms]);
```

```
    distOnArm    = fscanf(fid, '%f', [1,numArms]);
    psi          = fscanf(fid, '%f', [1,numArms]);
    distFScenter = fscanf(fid, '%f', 1);
    numViews     = fscanf(fid, '%f', 1);
    fclose(fid);
end

distFromTube = distFromTube + distFScenter;
distOnArm = -distOnArm;
numViews = round(numViews);
```

A.4 Filtering and Backprojection

A.4.1 Sinogram Filtering and Backprojection

```
function reconsImage = fast_iradon (sinogram, sizeRecons)
% Perform the sinogram filtering in frequency space, and call the
% fast inverse radon transform MEX file.


if nargin < 1,
    error ('Enter a sinogram.');
end

[nCells, nViews] = size(sinogram);

if nargin < 2,
    sizeRecons = nCells / sqrt(2);
end

filterSize = ceil(log2(nCells));
switch filterSize
    % The filters are currently loaded from .mat files. They should
    % be built on demand, independent of their size (nCells). That's
    % actually easy and fast, but first we must decide what type of
    % filter to use.
    case 12
        load Filter4096.mat;
        sinogram(end+1:8192,:) = 0;
    case 11
        load Filter2048.mat;
        sinogram(end+1:4096,:) = 0;
    case 10
        load Filter1024.mat;
        sinogram(end+1:2048,:) = 0;
    case 9
        load Filter512.mat;
        sinogram(end+1:1024,:) = 0;
```

```
    case 8
        load Filter256.mat;
        sinogram(end+1:512,:) = 0;
    case 7
        load Filter128.mat;
        sinogram(end+1:256,:) = 0;
    otherwise
        error (['Currently there is no support for ' ...
            num2str(nCells) ' parallel beams.']);
end

% Apply the filter in frequency space in one sweep. This is amazingly
% fast, but Matlab on Windows may have memory problems handling this
% for a large number of cells (probably not the case for 64 bit
% Matlab).
if ispc && size(sinogram,1) > 1000,
    for ii = 1:size(sinogram, 2),
        sinogram(:,ii) = fft(sinogram(:,ii)) .* freqFilter;
    end
    for ii = 1:size(sinogram, 2),
        sinogram(:,ii) = ifft(sinogram(:,ii));
    end
    filtSinog = real(sinogram);
    clear sinogram
else
    freqFilter = repmat(freqFilter, 1, nViews);
    filtSinog = real(ifft(fft(sinogram) .* freqFilter));
end
filtSinog(nCells+1:end, :) = [];

% f_iradon is a MEX funciton written in C. This is... slow (for big
% sizeRecons).
reconsImage = f_iradon(filtSinog, sizeRecons);
```

A.4.2 Fast Inverse Radon Transform for Backprojection

```
/*==================================================================
 * f_iradon.c: Very fast inverse radon transform tuned specifically
 *             for PowerPC processors.
 *================================================================*/

#include <math.h>
#include "mex.h"

/* Input Arguments */

#define SINOG       prhs[0]
#define REC_SIZE        prhs[1]


/* Output Arguments */
```

```
#define REC_IMG     plhs[0]

#define PI 3.14159265


static void backproject(
        double  reconsImage[],
        long    sizeRecons,
        double  sinog[],
        long    nCells,
        long    nViews)
{
    unsigned long ii, jj, kk;
    unsigned long imgInd, cellInd, sinogInd, kknCells;
    mxArray *X, *Y, *SP, *CP;
    double *x, *y, *xii, *yjj, *xend, *yend;
    double *reconsImgElem, *reconsImgEnd;
    double phi, sinPhi, cosPhi, cellsCenter, otherTerms;

    X  = mxCreateDoubleMatrix(sizeRecons, 1, mxREAL);
    Y  = mxCreateDoubleMatrix(sizeRecons, 1, mxREAL);
    SP = mxCreateDoubleMatrix(nViews, 1, mxREAL);
    CP = mxCreateDoubleMatrix(nViews, 1, mxREAL);

    x = mxGetPr(X);
    y = mxGetPr(Y);

    xend = x + sizeRecons;
    yend = y + sizeRecons;
//    sinPhi = mxGetPr(SP);
//    cosPhi = mxGetPr(CP);

    for (ii=0; ii < sizeRecons; ii++) {
        x[ii] = ii - (sizeRecons-1.0)/2.0;
    }
    for (jj=0; jj < sizeRecons; jj++) {
        y[jj] = jj - (sizeRecons-1.0)/2.0;
    }

    // The extra 0.5 in cellsCenter will save us some time since we
    // had to replace round(*) with floor(* + 0.5).
    cellsCenter = (nCells-1.0)/2.0 + 0.5;

        reconsImgEnd = reconsImage + sizeRecons*sizeRecons;
    reconsImgElem = reconsImage;
    while (reconsImgElem < reconsImgEnd) {
         *reconsImgElem++ = 0;
    }

    for (kk=0; kk < nViews; kk++) {
        phi = kk * 2*PI / nViews;
        sinPhi = -sin(phi);
        cosPhi = cos(phi);
```

```
        kknCells = kk*nCells;
        reconsImgElem = reconsImage;
                yjj = y;
        while (yjj < yend) {
            otherTerms = *yjj++ * cosPhi + cellsCenter;
                xii = x;
            while(xii < xend) {
                //cellInd = floor(x[ii]*sinPhi + otherTerms);
                //reconsImage[imgInd] += sinog[kknCells + cellInd];
                *reconsImgElem++ += sinog[kknCells +
                    (long) floor(*xii++ * sinPhi + otherTerms)];
                //imgInd++;
            }
        }
    }
}



void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray*prhs[] )
{
    double *reconsImage;
    double *sinog,*sizeRecons;
    long nCells,nViews;

    /* Check for proper number of arguments */

    if (nrhs != 2) {
        mexErrMsgTxt("Two input arguments required.");
    } else if (nlhs > 1) {
        mexErrMsgTxt("Too many output arguments.");
    }

    nCells = mxGetM(SINOG);
    nViews = mxGetN(SINOG);
    if (!mxIsDouble(SINOG) || mxIsComplex(SINOG)) {
           mexErrMsgTxt("Bad sinogram.");
    }

    sinog = mxGetPr(SINOG);
    sizeRecons = mxGetPr(REC_SIZE);

    /* Create a matrix for the return argument */
    REC_IMG =
         mxCreateDoubleMatrix(sizeRecons[0], sizeRecons[0], mxREAL);

    /* Assign pointers to the various parameters */
    reconsImage = mxGetPr(REC_IMG);

    /* Do the actual computations in a subroutine */
    backproject(reconsImage, sizeRecons[0], sinog, nCells, nViews);
    return;
```

```
}
```

A.5 Ring Correction Algorithm

```
function newImage = ring_corrector(varargin)
% Perform ring correction of a reconstructed image by dividing the
% image in wedges (like a pie), and each wedge in arcs. Within each
% wedge calculate the mean value of the pixels of each arc as the
% radius increases. Smooth the resulting curve with a median filter
% and use the difference between the two curves to adjust the pixels
% of each arc.

if nargin < 1 || isempty(varargin{1}),
    error('Provide an image to correct.');
else
    reconsImage = varargin{1};
end

if nargin < 2 || isempty(varargin{2}),
    numArcs = 16;
else
    numArcs = varargin{2};
end

if nargin < 3 || isempty(varargin{3}),
    medFilter = 15;
else
    medFilter = varargin{3};
end

if nargin < 4 || isempty(varargin{4}),
    deltaR = 1;
else
    deltaR = varargin{4};
end


[nrows, ncols] = size (reconsImage);
[xcoord, ycoord] = meshgrid(1:ncols,1:nrows);

if nargin < 5 || isempty(varargin{5}),
    xcoord = xcoord - (nrows+1)/2;
    ycoord = ycoord - (ncols+1)/2;
else
    xcoord = xcoord - varargin{5}(1);
    ycoord = ycoord - varargin{5}(2);
end

radii = sqrt(xcoord.^2 + ycoord.^2);
ringMembers = floor(radii / deltaR) + 1;
```

```
numRings = max(ringMembers(:));
angles = atan2(ycoord, xcoord) + pi;
angles(angles == 2*pi) = 0;
wedgeMembers = floor(angles / (2*pi/numArcs)) + 1;
arcMembers = (wedgeMembers-1)*numRings + ringMembers;

meanInten = zeros(numRings, numArcs);
countMem = zeros(numRings, numArcs);

for pixInd = 1:nrows*ncols,
    aM = arcMembers(pixInd);
    meanInten(aM) = meanInten(aM) + reconsImage(pixInd);
    countMem (aM) = countMem (aM) + 1;
end

countMem(countMem==0) = 1;
meanInten = meanInten ./ countMem;

smoothed = medfilt1(meanInten, medFilter);
corrections = (meanInten - smoothed)/2;

newImage = zeros(size(reconsImage));
newImage(:)=reconsImage(:)-corrections(arcMembers(:));

newImage(newImage == 0) = reconsImage(newImage == 0);
```

A.6 VRX Calibration

A.6.1 Calibration Program

```
function calibValues = vrx_calibration (calibParams,calibFileStyle)
% Determine the exact position of all the elements of the system
% (arms, center of rotation, focal spot of the x-ray tube) by
% fitting these parameters until the expected trajectory of the
% shadow of a rotating pin matches the measured sinogram.


% Get each input parameter from the calibParams structure
pinSinogFile    = calibParams.pinSinogFile;
pinPhase        = calibParams.pinPhase;
threshLevel     = calibParams.threshLevel;
pinToCOR        = calibParams.pinToCOR;
distFromCOR     = calibParams.distFromCOR;
distOnArm       = calibParams.distOnArm;
psi             = calibParams.psi;
distFScenter    = calibParams.distFScenter;
cycleViews      = calibParams.cycleViews;
vrxConfig       = calibParams.vrxConfig;
```

```
numCells        = calibParams.numCells;

if nargin < 2 || isempty(calibFileStyle)
    calibFileStyle = 'new';
end


distOnArm = -distOnArm;
[numArms, cellPositions] = get_configuration (vrxConfig);

% Record the future inter-arm gap indexes
gaps = (1:numArms)*size(cellPositions,1);


% Create the figures where the results will be plotted
global sinosFig armsFig errorValVec iterCount
sinosFig = gcf;
armsFig = figure;
pause(0.01);
errorValVec = [];
iterCount = 1;

pinSinog = get_sinogram(pinSinogFile, numCells, threshLevel);
calibViews = size(pinSinog, 2);

% phis: all the projection angles. Positive for clockwise, negative
% for counter-clocwise.
%phis = (0:calibViews-1)/cycleViews * 2*pi;
phis = -(0:calibViews-1)/cycleViews * 2*pi;

% Find centroids
cellIndexes = repmat((1:numCells)', 1, calibViews);
centroids = sum(cellIndexes.*pinSinog, 1)./sum(pinSinog,1);
centroids = centroids';

% Find the initial pinPhase (this overrides the value passed as a
% parameter)
pinPhase = find_pin_phase(centroids, numCells, cycleViews);

% Set up the vector of initial values for the minimization function
minVars = [pinPhase, distFromCOR, distOnArm, psi, distFScenter];
%minVars = [pinPhase, distFromCOR, distOnArm, psi, ...
%           distFScenter, pinToCOR];

minOp = optimset('Display', 'iter', 'MaxIter', 2000, ...
    'TolX', 1e-7, 'TolFun', 1e-7);
calibVars = fminsearch(@error_function, minVars, minOp, ...
    pinToCOR, cellPositions, gaps, centroids, phis);

pinPhase    = calibVars(1);
distFromCOR = calibVars(          2:  numArms+1);
distOnArm   =-calibVars(  numArms+2:2*numArms+1);
psi         = calibVars(2*numArms+2:3*numArms+1);
distFScenter= calibVars(end);
```

```
pinPhase*180/pi
distFromCOR
-distOnArm
disp([180-psi(1:ceil(end/2))*180/pi psi(floor(end/2)+1:end)*180/pi])
distFScenter

calibValues = struct( ...
    'distFromCOR',      distFromCOR, ...
    'distOnArm',        distOnArm, ...
    'psi',              psi, ...
    'distFScenter',     distFScenter, ...
    'cycleViews',       cycleViews);


if strcmp(calibFileStyle, 'old')
    % Use old style for calibration file (just a list of values)
    outValues = [distFromCOR distOnArm psi distFScenter cycleViews];
    save ([pinSinogFile, ' calib.dat'], 'outValues', '-ascii');
else
    % Use new style: put all the values in a structure
    save ([pinSinogFile, ' calib.mat'], 'calibValues');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Internal functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function pinSinog = get_sinogram(pinSinogFile, numCells, threshLevel)

if  strcmp(pinSinogFile(end-3:end), '.mat'),
    % The sinogram is in a .mat file, read the matrix directly.
    load(pinSinogFile);
    %sino = sino(:,1:1994);
    if strcmp(lower(calibParams.vrxConfig),'x-scan')
        fileContents = double(sino');
    else
        fileContents = double(flipud(sino'));
    end
else
    % Assume that the sinogram is in a text file
    % fileId = fopen(pinSinogFile, 'r');
    % [fileContents, nElem] =  fread(fileId, [numCells, inf], ...
    %                          'uint16',0,'l');
    % %[fileContents, nElem] =  fread(fileId, [numCells, inf], ...
    %                          'float32',0,'l');
    % fclose(fileId);
```

```
    % Small custom correction
    %fileContents(482,1029:1078) = 21744;

    % load (pinSinogFile)
    %fileContents = fliplr(double(sinog(:, 2:end-1)))';
    %fileContents = fliplr(sinog)';
    fid = fopen(pinSinogFile, 'r');
    inputSinog = fread(fid, [576,inf], 'float32', 0, 'l');
    fclose(fid);
    %inputSinog = flipud(inputSinog');
    fileContents = inputSinog;
end

pinSinog = mean(fileContents(:))./fileContents;
pinSinog(pinSinog < 1) = 1;
pinSinog = log(pinSinog);
pinSinog(pinSinog < threshLevel*max(pinSinog(:))) = 0;
%pinSinog(1:30,:) = 0;




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [numArms, cellPositions] = get_configuration (vrxConfig)
% Some info on the geometry of the VRX.

switch lower(vrxConfig)
    case '4 arms'
        numArms = 4;
        numCells = 144;                 % Num of cells per arm.
        interCellDist = .099;           % In cm.
        cellsPerMod = 24;               % Cells per module.
        interModDist = 2.402 - ...
            cellsPerMod*interCellDist;  % Extra gap between modules.
    case '2 arms'
        numArms = 2;
        numCells = 288;                 % Num of cells per arm.
        interCellDist = .099;           % In cm.
        cellsPerMod = 24;               % Cells per module.
        interModDist = 2.402 - ...
            cellsPerMod*interCellDist;  % Extra gap between modules.
    case 'x-scan'
        numArms = 1;
        numCells = 768;                 % Num of cells per arm.
        interCellDist = 32.72/numCells; % In cm.
        %interCellDist = 0.04; % In cm.
        cellsPerMod = 768;              % Cells per module.
        interModDist = 0;
    case 'fpd slice 1024'
        numArms = 1;
        numCells = 1020;                % Num of cells per arm.
```

```
        interCellDist = 0.0194;          % In cm.
        cellsPerMod = 1020;              % Cells per module.
        interModDist = 0;
    case 'fpd slice 512'
        numArms = 1;
        numCells = 508;                  % Num of cells per arm.
        interCellDist = 0.0194*2;        % In cm.
        cellsPerMod = 508;               % Cells per module.
        interModDist = 0;
end

% Build the cell positions relative to the first cell for a single
% arm. Note that we are considering an extra "cell" (as compared to
% the reconstruction algorithm). This extra element represents the
% end of the arm.
cellPositions = 0:(numCells);
cellPositions = cellPositions * interCellDist + ...
    floor(cellPositions/cellsPerMod) * interModDist;
% cellPositions has cell indexes in the rows, arms in the columns.
cellPositions = repmat(cellPositions', 1, numArms);









%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function errorVal = error_function (minVars, ...
    pinToCOR, cellPositions, gaps, centroids, phis)

global sinosFig armsFig errorValVec iterCount

[cellsPerArm,numArms] = size(cellPositions);
% There's a pseudo-cell attached at the end of each arm.
cellsPerArm = cellsPerArm - 1;

% Recover the variables from minVars
pinPhase    = minVars(1);
distFromCOR = minVars(          2:  numArms+1);
distOnArm   = minVars(  numArms+2:2*numArms+1);
psi         = minVars(2*numArms+2:3*numArms+1);
distFScenter= minVars(end);
%distFScenter= minVars(end-1); pinToCOR=minVars(end);


% Positions of the pin with the current config. Note that the "y"
% axis is inverted.
pinPosX = cos(phis + pinPhase)*pinToCOR + distFScenter;
pinPosY = sin(phis + pinPhase)*pinToCOR;
```

```
pinSlopes = pinPosY./pinPosX;
pinSlopes1 = pinSlopes;

% Find the equations for the arms
armSlopes = tan(psi);
armYinter = -armSlopes .* (distFScenter + distFromCOR);

% Find the intersects between the rays that pass through the pin,
% and the arms.
armSlopes = repmat(armSlopes, length(pinSlopes), 1);
armYinter = repmat(armYinter, length(pinSlopes), 1);
pinSlopes = repmat(pinSlopes', 1, numArms);
xIntersec = -armYinter ./ (armSlopes - pinSlopes);

% Project the xIntersec coordinate on the arm
xIntersec = repmat(distFScenter + distFromCOR, length(pinPosX), 1) ...
    ...
    - xIntersec;
pinOnArm = xIntersec ./ repmat(cos(psi), length(pinPosX), 1);

% If an arm is at psi = pi/2, we will have an error.
% Handle this case.
for armInd = 1:numArms,
    if psi(armInd) == pi/2,
        pinOnArm(:,armInd) = -pinSlopes1' * (distFScenter+distFromCOR...
            (armInd));
    end
end

% Adjust the displacement value (the arm origin is always on the
% leftmost cell)
pinOnArm = pinOnArm + repmat(distOnArm, length(pinPosX), 1);

% Find the cell that captures each pinOnArm on each arm (if any)
for armInd = 1:numArms,
    cellPos = repmat(cellPositions(:,armInd)', size(pinOnArm,1), 1);
    cellPos1 = cellPos(:, 1:end-1);
    cellPos2 = cellPos(:, 2:end);
    pinPos  = repmat(pinOnArm(:,armInd), 1, size(cellPos,2)-1);

    % Find the cells that capture the pin
    catchCells = (cellPos1 <= pinPos) & (pinPos < cellPos2);

    % Find the integer part of the index of the cell
    [dummy, integPart] = find(catchCells);

    % Find the fractionary part of the index of the cell
    fracPart = (pinPos(catchCells) - cellPos1(catchCells)) ...
        ./ (cellPos2(catchCells) - cellPos1(catchCells));

    % Find the total index of the cell
    pinPos = zeros(size(catchCells));
    pinPos(catchCells) = (integPart + fracPart) ...
        + (armInd-1)*cellsPerArm;
```

```
    % Concatenate the results for all arms in a single matrix
    pinCellIndex(armInd,:) = sum(pinPos');
end

% Invalidate indexes == 0; that means that the arm didn't see the
% pin
pinCellIndex(pinCellIndex == 0) = nan;


% If more than one arm sees the pin in a particular view, we are
% going to favor the leftmost one.
pinCellIndex = min(pinCellIndex, [], 1)';

% Find the sum of square errors between the expected pin signal and
% the centroids of the real sinogram.
sqErr = centroids - pinCellIndex;
% Penalize places where gaps don't match...
%sqErr( isnan(pinCellIndex) & ~isnan(centroids)) = 4;
%sqErr(~isnan(pinCellIndex) &  isnan(centroids)) = 4;
% ...but don't penalize when gaps do match
sqErr(isnan(sqErr)) = 0;
sqErr = sqErr.^2;
errorVal = sum(sqErr)/length(sqErr);
errorValVec = [errorValVec errorVal];

% Provide some feedback to the user
figure(sinosFig);
subplot(2,1,1); plot (1:length(centroids), centroids, ...
    1:length(centroids),pinCellIndex);
if mod(iterCount, 10) == 0,
    figure(armsFig);
    plot_vrx_arms (numArms, cellPositions, psi, ...
        distFScenter, distFromCOR, distOnArm);
    figure(sinosFig);
    subplot(2,1,2); plot(log10(errorValVec));
    title('Error function');
    xlabel('Function evaluation count');
    ylabel('log_{10}(error)');
end
iterCount = iterCount + 1;
pause(0.001);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function pinPhase = find_pin_phase(centroids, numCells, cycleViews)
% The pin crosses the central cell two or three times. At least one
% of them should be ascending (going from lef to right). We will use
% that crossing to determine the initial pinPhase.
```

```
%midCell = numCells/2;
midCell = (min(centroids)+max(centroids))/2;
[dummy, viewCandidates] = sort(abs(centroids-midCell));

% In principle, one of the first three candidates is the winner, but
% we take five for the unlikely case that the first four correspond
% to descending crossings that happen to be symmetrical around the
% central cell.
viewCandidates = viewCandidates(1:5);
indxCandidates = find (centroids(viewCandidates+1) ...
    - centroids(viewCandidates) > 0 );
phaseView = viewCandidates(indxCandidates(1));

%pinPhase = pi - 2*pi* phaseView/cycleViews;
pinPhase = 2*pi* phaseView/cycleViews;
```

A.6.2 Visualization of the System Calibration

```
function plot_vrx_arms (varargin)
%function plot_arms (armsFig, numArms, cellPositions, psi, ...
%    distFScenter, distFromCOR, distOnArm)
% Plot the arms and fields of view for verification purposes

[numArms, cellPositions, psi, distFScenter, ...
                distFromTube, distOnArm, numCells] ...
        = parse_inputs(varargin{:});

% Replicate arrays so that corresponding cells in cellPositions get
% the info for their arm.
psi = repmat (psi, numCells, 1);
distOnArm = repmat (distOnArm, numCells, 1);
distFromTube = repmat (distFromTube, numCells, 1);

% Find the angles (in radians) using sine/cosine laws and some
% algebra.
cellDistances = cellPositions - distOnArm;
cellX = distFromTube - cellDistances .* cos(psi);
cellY = cellDistances .* sin(psi);

%hold on
%line(cellX-distFScenter, -cellY, 'Color', [0 0 0], 'LineWidth', 4);
plot(cellX-distFScenter, -cellY,'.');
axis equal
if numArms == 4,
    % Use the first cells of the first two arms as the references
    % for the target and outer fields of view.
    bound(1) = 1;
    bound(2) = numCells+1;
else
    % Use the first and the last cell as the references for
    % the target and outer fields of view.
```

```
    bound(1) = 1;
    bound(2) = numCells*numArms;
    bound(3) = numCells;
end
boundX =  cellX(bound);
boundY = -cellY(bound);

circX = -boundY.^2*distFScenter ./ (boundX.^2 + boundY.^2);
circY = -boundX./boundY .* circX;
circR = sqrt(circX.^2 + circY.^2);

t = 0:.01:2*pi;
x = cos(t);
y = sin(t);

cc{1} = [0 0 1];
cc{2} = [0 1 0];
cc{3} = [1 0 0];
for ii = 1: length(bound),
    line([0, boundX(ii)]-distFScenter, [0, boundY(ii)], ...
        'Color', cc{ii}, 'LineWidth', 2);

    xx = x*circR(ii);
    yy = y*circR(ii);
    line(xx, yy, 'Color', cc{ii}, 'LineWidth', 2);
end
circR
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [numArms, cellPositions, psi, distFScenter, ...
                        distFromTube, distOnArm, numCells] ...
            = parse_inputs(varargin)

if numel(varargin) == 2
        % Use a calibration file.
        calibrationFile = varargin{1};
        vrxConfig = varargin{2};

        [numArms, cellPositions, dummy, numCells] ...
                = get_VRX_configuration (vrxConfig);
        [numViews, distFScenter, psi, distOnArm, distFromTube] ...
        = get_VRX_calibration (calibrationFile, numArms);
else
        % The input parameters are specified directly. Note that
        % distFromCOR is specified instead of distFromTube.
        numArms = varargin{1};
        cellPositions = varargin{2};
        psi = varargin{3};
        distFScenter = varargin{4};
```

```
        distFromCOR = varargin{5};
        distOnArm = varargin{6};

    distFromTube = distFScenter + distFromCOR;

        % cellPositions has an extra element at the end.
        cellPositions = cellPositions(1:end-1);
    [numCells, numArms] = size(cellPositions);
end
```

Appendix B

Multi-arm VRX Simulation Code

All the simulations described in this dissertation were made with the code listed in this appendix. The general simulation algorithm was described in Section 4.1 and was implemented in the program `simulate_VRX.m`, listed in Section B.1. As explained before, simulation of CT scanners is very computationally demanding, so the main bottlenecks were re-implemented in highly-optimzed C-Mex code, although the original MATLAB code is included also as it is far easier to understand. The code for the Mex functions is listed in Sections B.2. Other auxiliary programs are used to interface with SPECT (used to obtain the attenuation coefficients for the materials employed in the simulation) and to provide secondary functionality. These are listed in Section B.3.

B.1 Main Simulation Code

```
function [recImage, sinogram, matIntenMap] = simulate_VRX ( ...
    typeOfBeam, phantomFile, numViews, recSize, ...
    calibrationFile, vrxConfig, phantomTilt, hasMaterialPhant,...
       vrxAngle, ...
    raysPerCell, numEnergyBins)
% Simulation of a single-slice VRX scanner.
%

%-----------------------------------------------------------------
% Parameters. Distances in cm, times in sec.

global makePlots
if isempty (makePlots)
    makePlots = false;
end

if nargin < 1 || isempty(typeOfBeam),
    typeOfBeam = 'poly';
end

% Default phantom file
phantomDir = 'Phantoms/';
if nargin < 2 || isempty(phantomFile),
```

```
    phantomFile = 'Muscle, Bone, Fat, Air.circles';
end;

if nargin < 4 || isempty(recSize),
    % Defaults for reconstruction size: 1000 virtual rays,
    % 900x900 final image
    recSize = [1000,700];
end;

if nargin < 5 || isempty(calibrationFile),
    calibrationFile = './4_arm_simul_calib.dat';
    reconsCalibFile = './4_arm_simul_calib.dat';
else
    if ischar(calibrationFile) || isstruct(calibrationFile);
        % Reconstruct with the same calibration file
        reconsCalibFile = calibrationFile;
    else
        % Use different calibrations for simulation and
        % reconstruction
        reconsCalibFile = calibrationFile{2};
        calibrationFile = calibrationFile{1};
    end
end

if nargin < 6 || isempty(vrxConfig),
    vrxConfig = '4 arms';
end

if nargin < 7 || isempty(phantomTilt),
        % Unless specified, the pahntom isn't tilted initially
        phantomTilt = 0;
else
        % The tilt is given in degrees, but internally we work in
    % radians
        phantomTilt = phantomTilt * pi/180;
end

if nargin < 8 || isempty(hasMaterialPhant),
    hasMaterialPhant = false;
end

if nargin < 10 || isempty(raysPerCell),
    raysPerCell = 1;
end

if nargin < 11 || isempty(numEnergyBins),
    numEnergyBins = [];
end


% Get info on energy beam and phantom
%[materials, materialMat, densityMat, energySpectrum, ...
%    attenCoefs, materialDens] ...
%    = spect_interface ([], numEnergyBins);
```

```
load (['Spect_results/Spect_results_' num2str(numEnergyBins)]);

if ~strcmp(typeOfBeam, 'poly'),
    if ~isnumeric(typeOfBeam),
        typeOfBeam = 60;
    end
    monochromIndex = sum(energySpectrum(:,1) < typeOfBeam);
    energySpectrum(:,2) = 0;
    energySpectrum(monochromIndex,2) = 1;
end

% Get configuration of the VRX scanner
[numArms, cellPositions, cellWidth] ...
    = get_VRX_configuration (vrxConfig);
[numViewsCalib, distFScenter, psi, distOnArm, distFromTube] ...
    = get_VRX_calibration (calibrationFile, numArms);
%[numViewsCalib, distFScenter, psi, distOnArm, distFromTube] = ...
%    get_calibration ([], numArms);

if nargin < 3 || isempty(numViews),
    numViews = numViewsCalib;
end

% Generate the rays and find the points of intersection with the
% arms
[cellAngles, cell2FS, cellAngleCover, totalAngle] ...
    = fan_angles (cellPositions, cellWidth, psi, ...
    distOnArm, distFromTube, makePlots, distFScenter, raysPerCell);
cellAngles = -cellAngles(:);
cellSlopes = tan(cellAngles);
cell2FS = cell2FS(:);
anglePortions = cellAngleCover(:) / totalAngle;


% Find the total number of photons sent to the slice in each view.
% (At the time this is an arbitrary number).
photonsPerView = 1e11 / numViews;



% Read phantom file
[tissueType, tissueIndex, tissueCode, ...
    tissueCenter, tissueRadius, tissueTilt, tissueVertex] ...
    = read_phantom_file([phantomDir phantomFile]);

totViews = round(1.2 * numViews);
sinogram = zeros(totViews, numel(cellPositions));
newVertex = {};

progString = ['0 views of ' num2str(totViews)];
disp(progString)
tic
% Acquire each view
%for ii = 0:numViews-1,
```

```
for ii = 0:totViews-1,
    % Find the positions for the centers of the ellipses after
    % rotating and translating. (Rotate clockwise!)
    % Note that the phantom might have an initial tilt, and that
    % anyway we start scanning with an initial tilt that will be
    % taken into account when we clip the ends of the sinograms
    % during reconstruction.
    theta = 2*pi * (ii-round(0.1*numViews))/numViews - phantomTilt;
    rotMat = [cos(theta) -sin(theta); sin(theta) cos(theta)];
    newCenter = rotMat * tissueCenter;
    newCenter(1,:) = newCenter(1,:) + distFScenter;
    newTilt = tissueTilt + theta;

    % Find the vertices of the polygons after rotating and
    % translating.
    for vv = 1:length(tissueVertex),
        newVertex{vv} = rotMat * tissueVertex{vv};
        newVertex{vv}(1,:) = newVertex{vv}(1,:) + distFScenter;
    end

    % Plot the objects so we can verify the phantom visually.
    % (Uncomment only for testing/demos).
    %plot_phantom (tissueType, tissueIndex, tissueRadius, ...
    %    newCenter, newTilt, newVertex);

    % Find segments of all the rays traversing each disk (origin if
    % no intersection)
    [segmentLengths, segmentMatCodes] = ...
        find_ray_segments(cellSlopes, tissueType, tissueIndex, ...
            tissueCode, newCenter, tissueRadius, newTilt, newVertex);

    % Calculate the signal detected by each detector cell (ray)
    sinogView = calculate_signal (segmentLengths, ...
        segmentMatCodes, energySpectrum(:,2), attenCoefs);
    sinogView = signal_and_noise(sinogView, photonsPerView, ...
        anglePortions', cell2FS', raysPerCell);
    sinogram(ii+1,:) = sinogView;
    if mod(ii,20) == 0,
        lenProgS = length(progString);
        progString = [num2str(ii) ' views of ' num2str(totViews) ...
            '. Estimated time: ' num2str(toc*(totViews/(ii+1)-1))];
        disp([repmat(char(8), 1, lenProgS+1), progString]);
    end
end
toc

%sinogram = flipud(sinogram);

if makePlots
    imagesc(-log(sinogram));
    colormap gray
    pause(2);
end
```

```
% Reconstruct the image using the normal VRX reconstructor
recImage = reconstr_VRX_fan_angle(sinogram, ...
    recSize(1), recSize(2), reconsCalibFile, vrxConfig);
% recImage = fan_angle_reconstr(sinogram, recSize(1), recSize(2), ...
%     '../Jordan/Four_arms_VRX/caldelxdata.dat', '4 arms');
recImage = flipud(rot90(recImage));

if makePlots,
    figure
    imagesc(recImage);
    colormap gray
    axis image
end

if hasMaterialPhant
    % Generate a map of the "effective" attenuation coefficients for
    % each of the pixels in the reconstructed image
    matIntenMap = get_atten_map (phantomFile, recSize, ...
        distFScenter, [cellAngles(1), cellAngles(end)], ...
        energySpectrum, attenCoefs, materialDens, ...
        tissueType, tissueIndex, tissueCode, ...
        tissueCenter, tissueRadius, tissueTilt, ...
        tissueVertex, vrxAngle);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Internal functions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [tissueType, tissueIndex, tissueCode, ...
    tissueCenter, tissueRadius, tissueTilt, tissueVertex] ...
    = read_phantom_file(phantomFile)
% The file describes possibly overlapping disks and polygons of
% different materials. The objects are "stacked" in the order they
% appear in the file.

dotsInName = strfind(phantomFile, '.');
if isempty(dotsInName),
    error('I can only read .circles, .ellipses and .polyellip files!!...
        ');
end

switch phantomFile(dotsInName(end):end)
    case '.circles',
        % The original version could only have circles. They are now
        % treated as a particular case of ellipses.
        numAttributes = 4;
```

```
    fid = fopen(phantomFile, 'r');
    [fileContents,itemsRead] = fscanf(fid, '%f', ...
        [numAttributes,inf]);
    fclose(fid);

    if mod(itemsRead, numAttributes) ~= 0,
        error(['All rows in the phantom should have ' ...
            num2str(numAttributes) ' columns.']);
        % Note that internally we work with the matrix transposed
    end

    tissueCode   = fileContents (1,  :);
    tissueCenter = fileContents (2:3,:);
    tissueRadius = fileContents (4,  :);
    tissueRadius = repmat(tissueRadius, 2, 1);
    tissueTilt = zeros(size(tissueCode));

    % Identify all the elements as ellipses, specify no polygons.
    tissueType = ones(size(tissueCode));
    tissueIndex = 1:length(tissueCode);
    tissueVertex = {};

case '.ellipses',
    % A newer version accepts arbitrary ellipses only.
    % Each line contains: the material code (a number), the
    % coordinates of the center, both radii of the ellipse, and
    % the initial tilt of the ellipse.
    numAttributes = 6;

    fid = fopen(phantomFile, 'r');
    [fileContents,itemsRead] = fscanf(fid, '%f', ...
        [numAttributes,inf]);
    fclose(fid);

    if mod(itemsRead, numAttributes) ~= 0,
        error(['All rows in the phantom should have ' ...
            num2str(numAttributes) ' columns.']);
        % Note that internally we work with the matrix transposed
    end

    tissueCode   = fileContents (1,  :);
    tissueCenter = fileContents (2:3,:);
    tissueRadius = fileContents (4:5,:);
    tissueTilt   = fileContents (6,  :)*pi/180;

    % Identify all the elements as ellipses, specify no polygons.
    tissueType = ones(size(tissueCode));
    tissueIndex = 1:length(tissueCode);
    tissueVertex = {};

case '.polyellip',
    % The current version supports both ellipses and polygons.
    % The first number in each line is 1 if the line describes an
    % ellipse, 2 for polygon.
```

```
% If it's a polygon, the third number represents the number
% of vertices, and the coordinates of those vertices are
% listed, clockwise. The second number is always the material
% code.
% The rest of the line for ellipses is as described above.

tissueType = [];
tissueCode = [];
tissueCenter = [];
tissueRadius = [];
tissueTilt = [];
tissueVertex = {};

fid = fopen(phantomFile, 'r');
tn = 0;
ellipseIndex = 0;
polygonIndex = 0;

while ~feof(fid),
    tn = tn + 1;
    % Read a line of the file. First find out the type of
    % object and the material.
    tissueType(tn) = fscanf(fid, '%f', 1);
    tissueCode(tn) = fscanf(fid, '%f', 1);

    switch tissueType(tn),
        case 1,
            % This is an ellipse.
            ellipseIndex = ellipseIndex + 1;
            tissueCenter(:,ellipseIndex) ...
                = fscanf(fid, '%f', [2,1]);
            tissueRadius(:,ellipseIndex) ...
                = fscanf(fid, '%f', [2,1]);
            tissueTilt  (:,ellipseIndex) ...
                = fscanf(fid, '%f', 1)*pi/180;;
            tissueIndex(tn) = ellipseIndex;

        case 2,
            % This is a polygon
            polygonIndex = polygonIndex + 1;
            numVertices = fscanf(fid, '%f', 1);
            tissueVertex{polygonIndex} ...
                = fscanf(fid, '%f', [2,numVertices]);
            tissueIndex(tn) = polygonIndex;

        otherwise,
            error('I can only work with ellipses and polygons...
               !');
    end
end

fclose(fid);

% The last vertex of each polygon must be connected to the
```

```
        % first.
        for poli = 1:polygonIndex,
            tissueVertex{poli} = [tissueVertex{poli} ...
                tissueVertex{poli}(:,1)];
        end

        % Put the tissue indexes of the polygons after those of the
        % ellipses.
        tissueIndex(tissueType == 2) ...
            = tissueIndex(tissueType == 2) + ellipseIndex;
    otherwise,
        error('I can only read .circles, .ellipses and .polyellip ...
           files!!');
end

global makePlots
if isempty (makePlots)
    makePlots = false;
end
if makePlots,
    % Plot the objects so we can verify the phantom visually
    figure;
    plot_phantom (tissueType, tissueIndex, tissueRadius, ...
        tissueCenter, tissueTilt, tissueVertex);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function plot_phantom (tissueType, tissueIndex, tissueRadius, ...
    tissueCenter, tissueTilt, tissueVertex)
% Plot the objects so we can verify the phantom visually

t = 0:.01:2*pi;
x = cos(t);
y = sin(t);
numEllipses = length(tissueTilt);

plot(tissueCenter(1,1),tissueCenter(2,1),'w')
axis equal;
hold on
for jj = 1:length(tissueType),
    ii = tissueIndex(jj);
    if tissueType(jj) == 1, % disc/ellipse
        x1 = x*tissueRadius(1,ii);
        y1 = y*tissueRadius(2,ii);
        xx =  x1*cos(tissueTilt(ii)) - y1*sin(tissueTilt(ii)) ...
            +  tissueCenter(1,ii);
        yy =  x1*sin(tissueTilt(ii)) + y1*cos(tissueTilt(ii)) ...
            +  tissueCenter(2,ii);
        plot (xx,yy);
    elseif tissueType(jj) == 2, % polygon
        plot (tissueVertex{ii-numEllipses}(1,:), ...
```

```
            tissueVertex{ii-numEllipses}(2,:));
    else
        % We shouldn't get here because this error was handled
        % before.
        error('I can only work with ellipses and polygons!');
    end
end
hold off
pause(0.01)
figure



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [segmentLengths, segmentMatCodes] = ...
    find_ray_segments(raySlopes, tissueType, tissueIndex, ...
        tissueCode, tissueCenter, tissueRadius, tissueTilt, ...
        tissueVertex)
% Each ray that travels from the focal spot to the detectors will
% traverse several materials. Divide each ray in segments of one
% material each, and find the length of each segment.
%
% Will will do it in three phases:
% PHASE I: calculate the intersection points of each ray with the
%            ellipses.
% PHASE II: calculate the intersection points for the polygons.
% PHASE III: combine all the intersection points, sort them, find the
%            segments.

numRays = length(raySlopes);
numEllipses = length(tissueTilt);
numPolygons = length(tissueVertex);

%%% PHASE I: Intersections with the ellipses

% Solve *many* quadratic equations.

ra2 = tissueRadius(1,:).^2;
rb2 = tissueRadius(2,:).^2;
cosTilt = cos(tissueTilt);
sinTilt = sin(tissueTilt);

C1 = repmat(rb2.*cosTilt.^2 + ra2.*sinTilt.^2, numRays,1);
C2 = repmat(rb2.*sinTilt.^2 + ra2.*cosTilt.^2, numRays,1);
C3 = repmat(2 * cosTilt .* sinTilt .* (rb2 - ra2), numRays,1);
C4 = repmat(ra2 .* rb2, numRays,1);
x0 = repmat(tissueCenter(1,:), numRays,1);
y0 = repmat(tissueCenter(2,:), numRays,1);
m  = repmat(raySlopes, 1, length(tissueTilt));

a = C1 + m.^2.*C2 + m.*C3;
b = -2*x0.*C1 - 2*m.*y0.*C2 - (y0 + m.*x0).*C3;
```

```
c = (x0.^2.*C1 + y0.^2.*C2 + x0.*y0.*C3 - C4);

% a = repmat(raySlopes.^2 + 1, 1, length(tissueRadius));
% b = -2*(repmat(tissueCenter(1,:),length(raySlopes),1) ...
%       + raySlopes*tissueCenter(2,:));
% c = repmat(sum(tissueCenter.^2) - tissueRadius.^2, ...
%       length(raySlopes), 1);

theSqRoot = sqrt(b.^2 - 4*a.*c);
xIntersect1 = (-b + theSqRoot) ./ (2*a);
xIntersect2 = (-b - theSqRoot) ./ (2*a);

% Complex solutions don't intersect. Send them all to the origin.
realOnes = abs(imag(xIntersect1)) == 0;
xIntersect1 = xIntersect1 .* realOnes;
xIntersect2 = xIntersect2 .* realOnes;



%%% PHASE II: Intersections with the polygons.
xIntersect3 = zeros(numRays, 2*numPolygons);
warning('off', 'Matlab:divideByZero');
for poly = 1:numPolygons,
    x1 = repmat(tissueVertex{poly}(1,1:end-1), numRays, 1);
    y1 = repmat(tissueVertex{poly}(2,1:end-1), numRays, 1);
    x2 = repmat(tissueVertex{poly}(1,2:end), numRays, 1);
    y2 = repmat(tissueVertex{poly}(2,2:end), numRays, 1);
    m  = repmat(raySlopes, 1, size(x1,2));

    alpha = -(m.*x1 - y1) ./ (m.*(x2-x1) - (y2-y1));
    % The previous division is zero when the slope of the ray is the
    % same as the slope of the edge. Thus, they either overlap (and
    % we ignore their intersection), or they are parallel (and there
    % is no intersection).
    alpha(isinf(alpha)) = nan;
    % We are only interested in intersections that occur between the
    % two end points, i.e., where alpha is between 0 and 1. If it's
    % 0 then the intersection occurs at an endpoint, so ignore it and
    % consider the intersection with the adjacent edge (which should
    % have alpha==1).
    alpha(alpha < 0 | alpha >= 1) = nan;
    % Find the x coordinates of the intersections.
    x = x1 + alpha.*(x2-x1);
    % Discard non-intersecting edges for rays that do intersect the
    % polygon
    x = sort(x,2);
    xIntersect3(:,2*poly-1:2*poly) = x(:,1:2);
    % If the second (exit) point is NaN and the first (entry) point
    % isn't, then the ray just touched the polygon at a vertex.
    % Ignore the first intersection also.
    xIntersect3(isnan(xIntersect3(:,2)),1) = nan;
end
warning('on', 'Matlab:divideByZero');
% Send remaining non-intersections to the origin.
```

```
xIntersect3(isnan(xIntersect3)) = 0;



%%% PHASE III: Combine all the intersection points, sort them, find
%%%            the segments.

% Find the y coordinates of all intersection points, and calculate
% distances to the origin.
xIntersect = [xIntersect1, xIntersect2, xIntersect3];
yIntersect = repmat(raySlopes,1,2*(numEllipses+numPolygons)) ...
    .* xIntersect;
distIntersect = sqrt(xIntersect.^2 + yIntersect.^2)';

% Keep track of the object to which each intersection point
% corresponds
tissueIndex = sortrows([tissueIndex; 1:length(tissueIndex)]');
tissueIndex = tissueIndex(:,2);
% The int. pts. for ellipses are interlaced at the beginning of
% tissueIndex...
objectIndexes = repmat(tissueIndex(1:numEllipses), 2, 1);
% ... while the int. pts. for the polygons are placed by pairs
% at the end.
ellipseIndexes = repmat(tissueIndex(numEllipses+1:end)', 2, 1);
objectIndexes = [objectIndexes; ellipseIndexes(:)];
objectIndexes = repmat(objectIndexes, 1, size(distIntersect,2));

% For each ray, sort these distances, keeping track of the materials
% involved.
[sortedIntersect,sortIndex] = sort(distIntersect);
objectIndexes = objectIndexes(sortIndex);

if false,
    % This code was replaced...

    segmentMatCodes = zeros(size(distIntersect));
    for objectN = 1:length(tissueIndex),
        % In segmentMatCodes mark the interval that each ray
        % traverses through each object. Note that this may overlap
        % with boundaries of other objects.
        boundariesObject = find (objectIndexes == objectN);
        for ii = 1:2:length(boundariesObject),
            % Note that segmentMatCodes is being scanned as a vector
            % instead of a matrix. This is for speed, as this is a
            % major bottleneck.
%             segmentMatCodes(boundariesObject(ii): ...
%                 boundariesObject(ii+1)-1) ...
%                 = tissueCode(tissueIndex(objectN));
            segmentMatCodes(boundariesObject(ii): ...
                boundariesObject(ii+1)-1) ...
                = tissueCode(objectN);
        end
    end
else
```

```
    % ...by this MEX-function.

    segmentMatCodes = fast_material_marker ...
        (objectIndexes, tissueIndex, tissueCode);
end
segmentLengths = diff(sortedIntersect);
segmentMatCodes = segmentMatCodes(1:end-1,:);




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function raySignals = calculate_signal (segmentLengths, ...
        segmentMatCodes, energySpectrum, attenCoefs)

% A segmentMatCode of 0 means the corresponding segmentLength is
% outside the phantom. Map it to a new material, vacuum.
%    Add the new material:
attenCoefs = [attenCoefs, zeros(size(attenCoefs,1), 1)];
%    The new material code:
segmentMatCodes(segmentMatCodes==0) = size(attenCoefs, 2);

% For L (distances) and mu (lin. atten. coeff.),  1st dim is segment
% index, 2nd dim is energy bin, 3rd dim is ray index.
[numSegments, numRays] = size(segmentMatCodes);
numEbins = length(energySpectrum);

if false,
    % This code was replaced...

    L = repmat(reshape(segmentLengths,[numSegments,1,numRays]), ...
        [1, numEbins, 1]);
    segmentMatCodes2 = repmat(reshape(segmentMatCodes-1, ...
        [numSegments,1,numRays]), [1, numEbins, 1]);
    muIndexes = repmat(1:numEbins, [numSegments, 1, numRays]);
    muIndexes = segmentMatCodes2*numEbins + muIndexes;
    mu = attenCoefs(muIndexes);

    % For exponent and phi0, 1st dim is energy bin, 2nd dim is ray
    % index
    exponent = squeeze(-sum(mu .* L));
    phi0 = repmat(energySpectrum, 1, numRays);
    raySignals = sum(phi0 .* exp(exponent));
else
    % ...by this MEX-function.

    raySignals = fast_calc_signal (segmentLengths, ...
        segmentMatCodes-1, attenCoefs, numEbins, energySpectrum);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sinogView = signal_and_noise(sinogView, photonsPerView, ...
        anglePortions, cell2FS, raysPerCell)

% Find the number of photons that got detected by each cell.
sinogView = sinogView .* (photonsPerView * anglePortions);

if raysPerCell > 1,
    % Each cell is actually made up of raysPerCell pseudo-cells.
    sizeView = length(sinogView);
    sinogView = reshape(sinogView, raysPerCell, ...
        sizeView/raysPerCell);
    sinogView = sum(sinogView);
end

% Add the noise
%multiplier = 15;
%multiplier = 3;
multiplier = 0;
sinogView = sinogView ...
    + sqrt(sinogView).*randn(size(sinogView))*multiplier;
```

B.2 Optimized C code

B.2.1 Segments of Materials Traversed by an X-ray

```
/*=================================================================
 * fast_material_marker.c: Determines the order in which an x-ray
 *      beam reaches the interfaces bewtween objects/tissues.
 *
 *  Written by David Rendon, UTHSC, Memphis, Fall 2007
 *=================================================================*/

#include <math.h>
#include "mex.h"

/* Input Arguments */

#define    OBJECT_INDEXES    prhs[0]
#define    TISSUE_INDEX    prhs[1]
#define TISSUE_CODE        prhs[2]



/* Output Arguments */
```

```
#define    SEG_MAT_CODES      plhs[0]

#define PI 3.14159265


static void fill_mat_codes(
        double  segmentMatCodes[],
        double  objectIndexes[],
        double  tissueIndex[],
        double  tissueCode[],
        long    numSegments,
        long    numRays,
        long    numObjects)
{
    long objectN, objectInd, tissueCodeN, numObjIndexes;
    short foundObj;

    numObjIndexes = numSegments*numRays;

    for (objectN=0; objectN < numObjects; objectN++) {
        //tissueCodeN = tissueCode[(long)tissueIndex[objectN]-1];
        tissueCodeN = tissueCode[objectN];
        foundObj = 0;
        objectInd = 0;
        while (objectInd < numObjIndexes) {
            //if (objectIndexes[objectInd] != tissueIndex[objectN]) {
            if (objectIndexes[objectInd] != objectN+1) {
                // Do nothing
            }
            else if (foundObj == 0) {
                // Found the first intersection of a particular ray
                // with object objectN. Keep track of it.
                foundObj = 1;
            }
            else {
                // Found the second intersection, so stop tracking it
                foundObj = 0;
            }

            if (foundObj == 1) {
                // We are inside the object objectN, so register its
                // material in segmentMatCodes
                segmentMatCodes[objectInd] = tissueCodeN;
            }

            objectInd++;
        }
    }
}



void mexFunction( int nlhs, mxArray *plhs[],
```

```
          int nrhs, const mxArray*prhs[] )
{
    double *segmentMatCodes;
    double *objectIndexes, *tissueIndex, *tissueCode;
    long numSegments, numRays, numObjects;

    /* Check for proper number of arguments */

    if (nrhs != 3) {
        mexErrMsgTxt("Three input arguments required.");
    } else if (nlhs > 1) {
        mexErrMsgTxt("Too many output arguments.");
    }

    numSegments = mxGetM (OBJECT_INDEXES);
    numRays = mxGetN (OBJECT_INDEXES);
    numObjects = mxGetM (TISSUE_INDEX);

    objectIndexes = mxGetPr(OBJECT_INDEXES);
    tissueIndex = mxGetPr(TISSUE_INDEX);
    tissueCode = mxGetPr(TISSUE_CODE);

    /* Create a matrix for the return argument */
    SEG_MAT_CODES = mxCreateDoubleMatrix(numSegments, numRays,
            mxREAL);

    segmentMatCodes = mxGetPr(SEG_MAT_CODES);

    /* Do the actual computations in a subroutine */
    fill_mat_codes(segmentMatCodes, objectIndexes, tissueIndex,
            tissueCode, numSegments, numRays, numObjects);
    return;

}
```

B.2.2 Signal Acquired by the Detector Cells

```
/*==========================================================
 * fast_calc_signal.c: Calculates the signal contributed by each ray.
 *
 *  Written by David Rendon, UTHSC, Memphis, Fall 2007
 *==========================================================*/

#include <math.h>
#include "mex.h"

/* Input Arguments */

#define    SEGMENT_LENGTHS    prhs[0]
#define    SEG_MAT_CODES    prhs[1]
#define ATTEN_COEFS        prhs[2]
```

```
#define NUM_E_BINS        prhs[3]
#define ENERGY_SPECT    prhs[4]



/* Output Arguments */

#define    RAY_SIGNALS        plhs[0]

#define PI 3.14159265


static void calc_signal(
        double  raySignals[],
        double  segmentLengths[],
        double  segmentMatCodes[],
        double  attenCoefs[],
        long    numEbins,
        long    numSegments,
        long    numRays,
        double  energySpectrum[])
{
     long energyInd, rayInd, rayIndNumSeg;
    double *lengthInd, *lengthIndEnd, *matCodeInd;
    double mu, exponent;

    for (rayInd=0; rayInd < numRays; rayInd++) {
        raySignals[rayInd] = 0;

        rayIndNumSeg = rayInd * numSegments;
        for (energyInd=0; energyInd < numEbins; energyInd++) {
            exponent = 0;

            lengthInd = segmentLengths + rayIndNumSeg;
            lengthIndEnd = lengthInd + numSegments;

            matCodeInd = segmentMatCodes +  rayIndNumSeg;

            while (lengthInd < lengthIndEnd) {
                mu = attenCoefs[(long)(*matCodeInd++) * numEbins
                        + energyInd];
                exponent -= mu * *lengthInd++;
            }

            raySignals[rayInd] += energySpectrum[energyInd]
                    * exp(exponent);
        }
    }


}
```

```
void mexFunction( int nlhs, mxArray *plhs[],
          int nrhs, const mxArray*prhs[] )
{
    double *raySignals;
    double *segmentLengths, *segmentMatCodes, *attenCoefs,
           *numEbins, *energySpectrum;
    long numSegments, numRays;

    /* Check for proper number of arguments */

    if (nrhs != 5) {
        mexErrMsgTxt("Five input arguments required.");
    } else if (nlhs > 1) {
        mexErrMsgTxt("Too many output arguments.");
    }

    numSegments = mxGetM(SEG_MAT_CODES);
    numRays = mxGetN(SEG_MAT_CODES);

    segmentLengths = mxGetPr(SEGMENT_LENGTHS);
    segmentMatCodes = mxGetPr(SEG_MAT_CODES);
    attenCoefs = mxGetPr(ATTEN_COEFS);
    numEbins = mxGetPr(NUM_E_BINS);
    energySpectrum = mxGetPr(ENERGY_SPECT);

    /* Create a matrix for the return argument */
    RAY_SIGNALS = mxCreateDoubleMatrix(1, numRays, mxREAL);

    raySignals = mxGetPr(RAY_SIGNALS);

    /* Do the actual computations in a subroutine */
    calc_signal(raySignals, segmentLengths, segmentMatCodes,
            attenCoefs, numEbins[0], numSegments, numRays,
            energySpectrum);
    return;

}
```

B.3 Auxiliary Code

B.3.1 Interface With Spect

```
function [materials, materialMat, densityMat, energySpectrum, ...
    attenCoefs, materialDens] = spect_interface (inFile, ...
    numEnergyBins)
% Interface with the SPECT program.

if nargin < 1,
```

```
    % Assume that we don't want material/density matrices
    inFile = [];
end

% Read the files with attenuation coefficients
materials = {'air', 'bone', 'fat', 'muscle', 'water', 'Lead'};
materialCode = {'AI', 'BO', 'FA',  'MU',     'H2O1',  'PB'};
% A density of -1 means the material is a pre-defined mixture
materialDens = [ -1,   -1,   -1,    -1,       1,       -1];
% % Read the files with attenuation coefficients
% materials = {'air', 'bone', 'fat', 'muscle',  'Lead'};
% materialCode = {'AI', 'BO', 'FA',  'MU',       'PB'};
% % A density of -1 means the material is a pre-defined mixture
% materialDens = [ -1,   -1,   -1,    -1,        -1];
% % Read the files with attenuation coefficients
% materials = {'air', 'bone', 'fat', 'muscle',  'CsI',   'Lead', ...
%       'Copper'};
% materialCode = {'AI', 'BO', 'FA',  'MU',      'CS1I1', 'PB',   ...
%       'CU'};
% % A density of -1 means the material is a pre-defined mixture
% materialDens = [ -1,   -1,   -1,    -1,        4.51,    -1,    ...
%       -1];
% Read the files with attenuation coefficients
materials = {'air', 'bone'};
materials (2+(1:15)) = {'water'};
materialCode = {'AI', 'BO'};
materialCode (2+(1:15)) = {'H2O1'};
% A density of -1 means the material is a pre-defined mixture
materialDens = [ -1 , -1 ];
tempo = -7:7;
materialDens (2+(1:15)) = 1.1.^(tempo/7);

% Air and acrylic
materials = {'air', 'acrylic'};
materialCode = {'AI', 'C5O2H8'};
materialDens = [-1, 1.19];


spectDir = '../ITS+Spect/Spect_mod_copy/';
spectDir = '~/Documents/DiBianca/ITS+Spect/Spect_mod_copy/';

colorMats = ...
   [0  0   0;
    255 255 255;
    255 255 0;
    255 0   0;
    0   255 0;
    0   255 0;
    0   0   255];

% Get the attenuation coefficients for all the materials (runs Spect
% for each one).
[attenCoefs,materialDens] = attenuation_coefs(spectDir,...
    materials, materialCode, materialDens, numEnergyBins);
```

```
if isstr(inFile),
    % inFile contains the name of a phantom image. Produce
    % materialMat and densityMat.
    readMat = imread(inFile);
    materialMat = zeros(size(readMat,1), size(readMat,2));

    for ii=1:length(materials),
        materialMat(readMat(:,:,1)==colorMats(ii,1) ...
            & readMat(:,:,2)==colorMats(ii,2) ...
            & readMat(:,:,3)==colorMats(ii,3)) = ii;
    end

    densityMat = materialDens(materialMat);
else
    % We don't want material/density matrices, so assign them dummy
    % values.
    materialMat = [];
    densityMat = [];
end


if 1,       % true Spectrum?
    % Read the spectrum file left by the runs of Spect above.
        fid = fopen([spectDir 'tube_sp.txt'], 'r');
        for ii=1:10,
        fgetl(fid);                 % Skip header lines.
        end
        energySpectrum = fscanf(fid, '%f', [2,inf])';
        fclose(fid);
end

% Normalize the spectrum
energySpectrum(:,2) = energySpectrum(:,2)/sum(energySpectrum(:,2));
global makePlots
if isempty (makePlots)
    makePlots = false;
end
if makePlots
    figure;plot(energySpectrum(:,1), energySpectrum(:,2));
    title  ('Distribution of photons in incoming beam');
    xlabel ('Photon energy,   keV');
    ylabel ('PDF');
end
```

B.3.2 Obtain Attenuation Coefficients

```
function [attenCoefs, densities] = attenuation_coefs ...
    (spectDir, materials, materialCode, materialDens, numEnergyBins)
% Uses Spect to calculate the attenuation coefficients for all the
% materials, as well as the energy spectrum of the beam.
```

```
numMaterials = length(materials);

densities = zeros(1,numMaterials);
attenCoefs = [];
for ii=1:numMaterials,
    spectStatus = run_Spect (spectDir, ...
        materialCode{ii}, materialDens(ii), numEnergyBins);
    if spectStatus ~= 0,
        error('Spect:fail', 'Spect gave a failure exit code!');
    end
    fid = fopen([spectDir 'atncoef.txt'],'r');
    fscanf(fid, '%s', 1);
    checkCode = fscanf(fid,'%s',1);
    if ~strcmp(checkCode, materialCode{ii}),
        disp(['Warning: File for ' materialCode{ii} ...
            ' contains ' checkCode]);
    end
    fscanf(fid, '%s', 2);
    densities(ii) = fscanf(fid, '%f', 1);
    if materialDens(ii) ~= -1 && materialDens(ii) ~= densities(ii),
        disp(['Warning: Density of ' materials{ii} ...
                ' changed: was ' materialDens(ii) ...
                ', now is ' densities(ii)]);
    end
    attenCoefs = [attenCoefs; fscanf(fid, '%f', [3,inf])'];
    fclose(fid);
end

energies = attenCoefs(1:size(attenCoefs,1)/numMaterials, 1);
attenCoefs = attenCoefs(:,2);
attenCoefs = reshape(attenCoefs,length(attenCoefs)/numMaterials, ...
    numMaterials);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the attenuation coefficients
attenPlotType = 'none';               % Should be added to params?

switch attenPlotType,
    case 'linear',
        attenCoefsPlot = attenCoefs;
        energiesPlot = energies;
        xlabelPlot = 'Photon energy,   keV';
        ylabelPlot = '\mu,   cm^{2}/g';
    case 'mass',
        % Spect returns the linear attenuation coefficients, we want
        % the mass att.coef. for the plots (to compare with
        % references). Also, energies in MeV.
        attenCoefsPlot = attenCoefs ./ repmat(densities, ...
            length(energies), 1);
        energiesPlot = energies/1000;
        xlabelPlot = 'Photon energy,   MeV';
        ylabelPlot = '\mu / \rho,   cm^{2}/g';
```

```
    case 'none',
        % Used later to supress the plot if it isn't wanted
end

if ~strcmp(attenPlotType, 'none'),
    % for ii=1:numMaterials,
    %     %figure
    %     loglog(energiesPlot, attenCoefsPlot);
    %     title(fileNames{ii});
    % end
    % loglog(energiesPLot attenCoefs.*repmat(densities, ...
    %    length(energies), 1));
    loglog(energiesPlot, attenCoefsPlot);
    legend (materials);
    title('Attenuation coefficients for the materials used');
    xlabel(xlabelPlot);
    ylabel(ylabelPlot);
end
```

B.3.3 Run Spect Processes

```
function spectStatus = run_Spect ...
    (spectDir, materialCode, materialDens, numEnergyBins)
% Runs Spect. A material density of -1 means the material is a
% pre-defined mixture

if nargin < 4 || isempty(numEnergyBins),
    numEnergyBins = [];
    inputFile = 'interaction';
else
    inputFile = ['interaction_' num2str(numEnergyBins)];
end

spectExe = 'SpectMod.mac';

tubeType        = 1;                    % Thermionic
peakVoltage     = 80;  %120;                 % in kVp
tubeCharge      = 1;   %100;                 % in mA s
voltRipple      = 1;                    % 1%
currRipple      = 1;                    % 1%
%stepSize        = 1%0.20;                 % in keV
imageDist       = 100;                  % in cm
anodeAngle      = 10;                   % in degrees
intrinsicFilt   = 0.3;                  % in cm (tube+collimator)

if isempty (numEnergyBins),
    stepSize        = 1;
else
    stepSize = peakVoltage / numEnergyBins;
end
```

```
% Clean up the Spect dir
prevDir = pwd;
cd (spectDir);
delete atncoef.txt detr_sp.txt output.txt tube_sp.txt;

% Build CLI interaction file
fid = fopen(inputFile, 'w');
fprintf (fid, 'K\n');                    % "Keyboard" input
fprintf (fid, '%d\n', tubeType);
fprintf (fid, '%f\n', [peakVoltage tubeCharge voltRipple ...
        currRipple stepSize imageDist anodeAngle intrinsicFilt]);

fprintf (fid, 'Y\n');                    % Additional Absorption
if materialDens == -1,
    fprintf (fid, '3\n');                % A mixture
    fprintf (fid, '%s1\n1\n', materialCode); % Dummy factor,
                                             %dummy thickness
else
    fprintf (fid, '1\n');                % A compound
    fprintf (fid, '%s\n', materialCode);
    fprintf (fid, '%f\n1\n', materialDens); % Dummy thickness
end
fprintf (fid, 'F\n');                    % Absorber is filter
fprintf (fid, 'N\n');                    % No more absorption
fclose (fid);

% Run the thing
spectStatus = system(['./' spectExe ' < ' inputFile ' > salida']);

cd (prevDir);
```

B.3.4 Calculate the Fan Angle for the Detector Cells

```
function [cellAngles, cell2FS, cellAngleCover, totalAngle] ...
            = fan_angles (cellPositions, cellWidth, psi, ...
            distOnArm, distFromTube, makePlot, distFScenter, ...
            raysPerCell)
% Finds the fan angle at which each cell of a multi-arm single slice
% VRX scanner is located (relative to the focal spotÑcenter of
% rotation line).
%
% Angles in radians, negative for left arm, positive for right arm.
%
% cellPositions: distance in cm to each cell from reference point of
%   the arm. (columns represent the different arms).
% psi: angle formed by fs-cor line and each arm (in radians).
% distOnArm: distance to intersection of fs-cor line and each arm,
%   form reference point.
% distFromTube: distance from the intersections to the focal spot.
%
%
```

```
if nargin < 6 || isempty(makePlot),
    makePlot = 0;
end

if nargin < 8 || isempty(raysPerCell),
    raysPerCell = 1;
end

if raysPerCell > 1,
   % We need to replace each real cell with raysPerCell little
   % virtual cells.
   halfRange = cellWidth*(1 - 1/raysPerCell) / 2;
   miniCellOffsets = linspace(-halfRange, halfRange, raysPerCell);
   miniCellOffsets = repmat(miniCellOffsets', ...
       [1, size(cellPositions)]);
   cellPositions = reshape(cellPositions, [1, size(cellPositions)]);
   cellPositions = repmat(cellPositions, [raysPerCell, 1, 1]);
   cellPositions = cellPositions + miniCellOffsets;
   cellPositions = reshape(cellPositions, ...
       raysPerCell*size(cellPositions,2), size(cellPositions,3));
end

[numCells, numArms] = size(cellPositions);

% Replicate arrays so that corresponding cells in cellPositions get
% the info for their arm.
psi = repmat (psi, numCells, 1);
distOnArm = repmat (distOnArm, numCells, 1);
distFromTube = repmat (distFromTube, numCells, 1);

% Find the distances to each cell from the corresponding intersection
% point.
cellDistances = cellPositions - distOnArm;

% Find the angles (in radians) using sine/cosine laws and some
% algebra.
cellX = distFromTube - cellDistances .* cos(psi);
cellY = cellDistances .* sin(psi);
cell2FS = sqrt(cellX.^2 + cellY.^2);

cellAngles = atan(cellY ./ cellX);

% Repeat for the left and right boundaries of the cells to obtain
% the arcs subtended.
cellDistLeft = cellDistances - cellWidth/2;
cellXL = distFromTube - cellDistLeft .* cos(psi);
cellYL = cellDistLeft .* sin(psi);
cellAnglesLeft = atan(cellYL ./ cellXL);


cellDistRight = cellDistances + cellWidth/2;
cellXR = distFromTube - cellDistRight .* cos(psi);
cellYR = cellDistRight .* sin(psi);
```

```
cellAnglesRight = atan(cellYR ./ cellXR);

cellAngleCover = cellAnglesRight - cellAnglesLeft;
totalAngle = cellAnglesRight(end) - cellAnglesLeft(1);



if makePlot,
    % Plot the arms and fields of view for verification purposes
    figure
    hold on
    plot(cellX-distFScenter, -cellY,'.');
    axis equal
    if numArms == 4,
        % Use the first cells of the first two arm as the references
        % for the target and outer fields of view.
        bound(1) = 1;
        bound(2) = numCells+1;
    else
        % Use the first and the last cell as the references for
        % the target and outer fields of view.
        bound(1) = 1;
        bound(2) = numCells*numArms;
        bound(3) = numCells;
    end
    boundX =  cellX(bound);
    boundY = -cellY(bound);

    circX = -boundY.^2*distFScenter ./ (boundX.^2 + boundY.^2);
    circY = -boundX./boundY .* circX;
    circR = sqrt(circX.^2 + circY.^2);

    t = 0:.01:2*pi;
    x = cos(t);
    y = sin(t);

    cc{1} = [0 0 1];
    cc{2} = [1 0 0];
    cc{3} = [0 0 1];
    for ii = 1: length(bound),
        line([0, boundX(ii)]-distFScenter, [0, boundY(ii)], ...
            'Color', cc{ii}, 'LineWidth', 2);

        xx = x*circR(ii);
        yy = y*circR(ii);
        line(xx, yy, 'Color', cc{ii}, 'LineWidth', 2);
    end
    circR
end
```

B.3.5 Map of Effective Attenuations

```
function [matIntenMap] = get_atten_map ( ...
    phantomFile, recSize, distFScenter, limitAngles,...
    energySpectrum, attenCoefs, materialDens, ...
    tissueType, tissueIndex, tissueCode, ...
    tissueCenter, tissueRadius, tissueTilt, tissueVertex, vrxAngle)
% Generate a map of the "effective" attenuation coefficients for
% each of the pixels in the reconstructed image. Also a map with the
% attenuationsat the "effective monochromatic energy", and a map of
% densities.

% Find the diameter of the total FOV
halfTotalAngle = max(abs(limitAngles));
diameterFOV = 2 * distFScenter * sin(halfTotalAngle);

% Find the length of the side of the reconstruction matrix
recSide = recSize(2)/recSize(1) * diameterFOV;

% Generate matrices with the coordinates of the pixels in the matrix
% Initially we will biuld a matrix that is larger than the
% reconstruction matrix and at the end we will scale the resulting
% matrices to the correct size.
scaleFactor = 2;
coords = linspace(-recSide/2, recSide/2, ...
    round(scaleFactor*recSize(2)));
[xMat, yMat] = meshgrid(coords, -coords);



% Add a new material, vacuum.
attenCoefs = [attenCoefs, zeros(size(attenCoefs,1), 1)];
materialDens = [materialDens, 0];



% Calculate an "effective monochromatic energy", and estimate the
% attenuation coefficient of each material at that energy.
monochromEnergy = sum(energySpectrum(:,1) .* energySpectrum(:,2)) ...
    / sum(energySpectrum(:,2));
prevEnergy = max(find(energySpectrum(:,1) <= monochromEnergy));
nextEnergy = min(find(energySpectrum(:,1) >= monochromEnergy));
if prevEnergy == nextEnergy,
    % The "effective energy" coincides with one of the table
    monochromeAttenCoefs = attenCoefs(prevEnergy, :);
else
    monochromeAttenCoefs = ( attenCoefs(prevEnergy,:) * ...
            (monochromEnergy - energySpectrum(prevEnergy, 1)) + ...
        attenCoefs(nextEnergy,:) * ...
            (energySpectrum(nextEnergy, 1) - monochromEnergy) ) ...
        / (nextEnergy - prevEnergy);
end
```

```
% For each material, calculate an "effective attenuation coefficient"
energySpectMatrix = repmat(energySpectrum(:,2), 1, ...
    size(attenCoefs,2));
effectAttenCoefs = sum(energySpectMatrix .* attenCoefs, 1) / ...
    sum(energySpectrum(:,2));



% Build the matrix with material codes (the codes will be used to
% generate all the output matrices). Start with all vacuum.
codeMat = zeros(size(xMat)) + length(materialDens);

numEllipses = length(tissueTilt);
for objectN = 1:length(tissueIndex),
    % Replace all pixels in the interior of object objectN with
    % tissueIndex(objectN)
    ii = tissueIndex(objectN);
    if tissueType(objectN) == 1, % disc/ellipse
        xM = xMat - tissueCenter(1,ii);
        yM = yMat - tissueCenter(2,ii);
        xx =  xM*cos(tissueTilt(ii)) + yM*sin(tissueTilt(ii));
        yy = -xM*sin(tissueTilt(ii)) + yM*cos(tissueTilt(ii));
        funcMat = (xx./tissueRadius(1,ii)).^2 + ...
            (yy./tissueRadius(2,ii)).^2;
        codeMat(funcMat < 1) = tissueCode(ii);
    elseif tissueType(objectN) == 2, % polygon
        maskMat = true(size(codeMat));
        vertVects = tissueVertex{ii-numEllipses}(:,2:end) ...
            - tissueVertex{ii-numEllipses}(:,1:end-1);
        for vv = 1:size(vertVects, 2),
            % Zero out the pixels in funcMat that are out of the
            % polygon as determined by vertices vv and vv+1.
            % The polygon edge is defined clockwise.
            xM = xMat - tissueVertex{ii-numEllipses}(1,vv);
            yM = yMat - tissueVertex{ii-numEllipses}(2,vv);
            if vertVects(1,vv) == 0,
                % The vertices determine a vertical edge.
                if vertVects(2,vv) > 0,
                    % Vector pointing up
                    maskMat (xM <= 0) = false;
                else
                    maskMat (0 <= xM) = false;
                end
            else
                funcMat = yM - vertVects(2,vv)/vertVects(1,vv) * xM;
                if vertVects(1,vv) < 0,
                    % The vector is pointing left
                    maskMat (funcMat <= 0) = false;
                else
                    % The vector is pointing right
                    maskMat (funcMat >= 0) = false;
```

```
                end
            end
        end
        codeMat(maskMat) = tissueCode(ii);
    else
        % We shouldn't get here because this error was handled before
        error('I can only work with ellipses and polygons!');
    end
end

% % Build the map of densities
% densityMap = imresize(materialDens(codeMat), ...
%               [recSize(2),recSize(2)]);
%
% % Build map of attenuation coefficients for the equivalent
% % monochromatic beam.
% monoMap = imresize(monochromeAttenCoefs(codeMat), ...
%       [recSize(2),recSize(2)]);
%
% % Build map of effective attenuation coefficient (weighted by
% % energy spectrum)
% attenMap = imresize(effectAttenCoefs(codeMat), ...
%       [recSize(2),recSize(2)]);


% Build a map of target values based on material calibration phantom
load (['Results/Mater_calib_' sprintf('%0.1f',vrxAngle) '.mat']);
materialInten = [materialInten min(materialInten)];
%matIntenMap = imresize(materialInten(codeMat), ...
%       [recSize(2),recSize(2)], 'bilinear');
matIntenMap = new_resize(materialInten(codeMat));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function newImage = new_resize(origImage);

newImage = (origImage(1:2:end,1:2:end) ...
    +  origImage(2:2:end,1:2:end) ...
    +  origImage(1:2:end,2:2:end) ...
    +  origImage(2:2:end,2:2:end))/4;
```

Vita

David Alejandro Rendon was born in 1973 in Bogotá, Colombia. In 1999 he graduated from Los Andes University with Bachelor of Science degrees in Mathematics and in Electrical Engineering. After working for a few years in the telecommunications industry in Colombia, he decided to pursue a career in the Biomedical Imaging field. For this, he entered the Joint Program in Biomedical Engineering of the University of Tennessee Health Science Center and the University of Memphis, in Memphis, TN, where he received his Master of Science degree in 2005 and continued his doctoral studies. His research involved different aspects related to bioimaging, in particular the development of computational tools for the acquisition, processing and analysis of biomedical images, with emphasis on experimental computer tomography devices.