2014

Early-state damage detection, characterization, and evolution using high-resolution computed tomography

Robert Grandin
Iowa State University

Early-state damage detection, characterization, and evolution using high-resolution computed tomography

by

Robert John Grandin

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Engineering Mechanics

Program of Study Committee:

Joseph Gray, Co-major Professor

Thomas Rudolphi, Co-major Professor

James Evans

Stephen Holland

Peter Sherman

Richard Wlezien

Iowa State University

Ames, Iowa

2014

DEDICATION

To my wife, Laura. None of this would have been possible without your love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take the opportunity to recognize several individuals who have been of great help while working on this project.

First and foremost my major professors, Drs. Joe Gray and Thomas Rudolphi. Your guidance has been of immense value.

Additionally, I would like to thank Dr. Terry Jensen and Scott Wendt for all of their help, particularly while I was learning to use the various lab systems and troubleshooting the odd equipment error.

I would like to also thank several other CNDE staff and students, Dan Barnard, Darrel Enyart, Sunil Chakrapani, and Ajith Subramanian, for their help in obtaining samples.

This material is based on work supported by the Army Research Laboratory as part of cooperative agreement number W911NF0820036 at the Center for Nondestructive Evaluation at Iowa State University.

ABSTRACT

Safely using materials in high performance applications requires adequately understanding the mechanisms which control the nucleation and evolution of damage. Most of a material's operational life is spent in a state with noncritical damage, and, for example in metals only a small portion of its life falls within the classical Paris Law regime of crack growth. Developing proper structural health and prognosis models requires understanding the behavior of damage in these early stages within the material's life, and this early-stage damage occurs on length scales at which the material may be considered "granular" in the sense that the discrete regions which comprise the whole are large enough to require special consideration.

Material performance depends upon the characteristics of the granules themselves as well as the interfaces between granules. As a result, properly studying early-stage damage in complex, granular materials requires a means to characterize changes in the granules and interfaces. The granular-scale can range from tenths of microns in ceramics, to single microns in fiber-reinforced composites, to tens of millimeters in concrete. The difficulty of direct-study is often overcome by exhaustive testing of macro-scale damage caused by gross material loads and abuse. Such testing, for example optical or electron microscopy, destructive and further, is costly when used to study the evolution of damage within a material and often limits the study to a few snapshots. New developments in high-resolution computed tomography (HRCT) provide the necessary spatial resolution to directly image the granule length-scale of many materials. Successful application of HRCT with fiber-reinforced composites, however, requires extending the HRCT performance beyond current limits. This dissertation will discuss improvements made in the field of CT reconstruction which enable resolutions to be pushed to the point of being able to image the fiber-scale damage structures and the application of this new capability to the study of early-stage damage.

CHAPTER 1. INTRODUCTION

Meeting increasing demands on the performance of engineering systems and structures requires, as a key component of that goal, the development and application of new materials. Before such materials may be integrated into an engineering solution, their mechanical behavior must be characterized and adequately understood. Achieving this understanding includes several aspects, including the determination of typical elastic mechanical properties (e.g., Young's modulus, Poisson's ratio, plasticity) as well as studying the mechanisms by which damage is initiated and evolved. While these standard engineering treatments of materials has been very successful, interest in extending the life of structures such as airplanes and nuclear power plants, coupled with improving the performance of systems, is driving interest in understanding the early health state of components within a system. To understand early stages of damage, we have to account for a complexity in materials in that they are made from discrete components or gains. When this need is coupled with the normal new-material qualification process, means to to characterize materials at a granular length scale are required. The granular length scale refers to a size where the constituents of a material control its behavior; in metals it is the grain size and grain boundaries, in composites it is the fiber diameter and bond between fiber and matrix, in concrete it is the gravel and aggregate and the interface with the binding cement.

In many applications, particularly safety critical industries, such as aerospace, this process of characterizing materials requires a significant amount of time and money before a material is approved for use. Complex materials, such as fiber-reinforced composites, are not adequately described by the classical elastic parameters and require even more-involved study. Many complex materials may be considered "granular" in the sense that the discrete regions which comprise the whole are large enough to require special consideration. Material performance depends upon the characteristics of the interfaces between granules, and these interfaces are often the source of early-stage damage. As a result, properly studying early-stage damage in complex, granular materials requires a means to characterize the granule interfaces.

Damage Quantification

This need to study early-stage damage is driven by the desire to quantify the remaining time before a component must be replaced, a task typically referred to as “prognosis”. There are several inter-related sub-tasks on which prognosis depends, shown in figure 1.1. An obvious consideration is the current state of the structure, captured by the “Damage Quantification” sub-task. For most of a structure’s life, any cracks, or other damage, will be very small. The length scales involved are typically on the same order as the material granularity itself, and in many materials this is a sufficiently-small scale so as to make direct measurements difficult. Directly measuring material properties on the granular scale has often required optical or electron microscopy, which impose significant time and cost limitations in addition to interfering with the material’s response to applied loads as a result of mechanically preparing the material surface for microscopy. As shown in table 1.1, the granular-scale can range from tenths of microns in ceramics, to single microns in fiber-reinforced composites, to tens of millimeters in concrete.

Table 1.1 Granularity properties for selected materials.

Material	**Grain Volume Fraction**	**Length Scale**	**Grain Shape**
Concrete	0.75	500 μm - 5000 μm	Irregular Volumetric
Soils	0.1 - 0.9	0.1 μm - 500 μm	Volumetric
Metals	0.95	0.1 μm - 100 μm	Volumetric
Epoxy-Matrix Composites	0.5	0.1 μm - 10 μm	Needle
Ceramic-Matrix Composites	0.5 - 0.9	0.05 μm - 10 μm	Needle

Sensor Response

An additional sub-task required for prognosis is understanding the “Sensor Response” obtained while monitoring the state of the structure. There is often an inverse correlation between the clarity or specificity of a signal and the cost of measuring that signal. For example, pas-

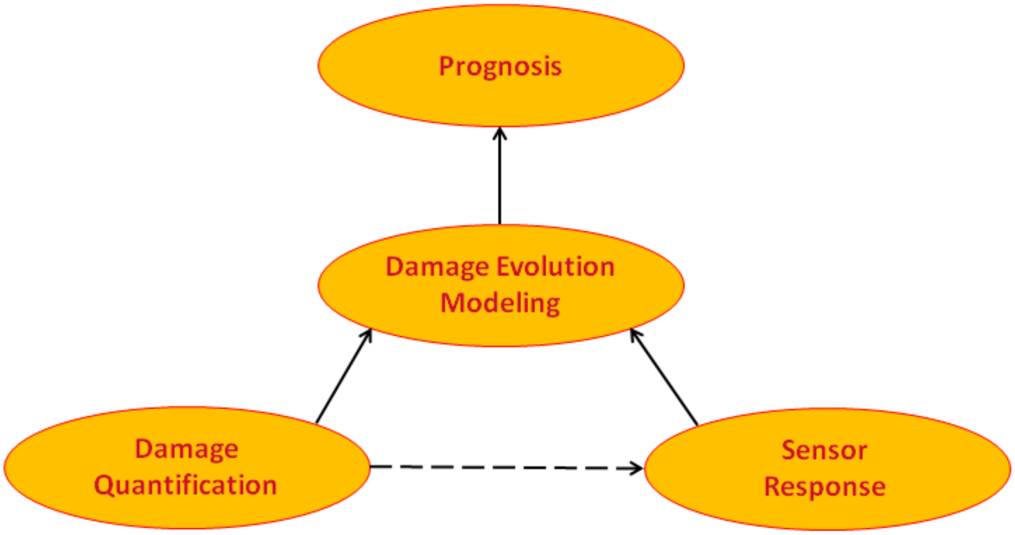


Figure 1.1 Components of prognosis. Performing prognosis requires a detailed understanding of the NDE signals produced by various sensors, development of damage-evolution models, and quantification of the current damage state. These components are in-turn inter-dependent upon each other, forming a dependency web which must be satisfied in order to achieve the desired end-goal of determining the remaining service life. The solid black arrows indicate direct inputs from one task to another, such as the damage evolution models requiring input which quantifies the current damage. The dashed arrow indicates a "training" input where the high-resolution laboratory techniques used for damage quantification are used to develop a proper interpretation of signals from economically-priced sensors which may be used outside of the laboratory environment.

sively listening for acoustic events is very low-cost but the resulting signal offers little more than a notification that some type of event occurred. A more-involved technique, such as an ultrasonic or eddy-current C-scan, can provide location information but often not at the granular resolution. Visual inspections, such as microscopy, can provide the necessary resolution but are limited to surface observations. High-resolution laboratory techniques, such as microscopy or computed tomography, can be used to determine relevant signatures produced by lower-cost sensors used outside the laboratory. Determination of signatures allows inspectors to determine what features are present within the material as well as what features cannot be present.

Damage Evolution Modeling

The third sub-task required for prognosis is "Damage Evolution Modeling". It is not sufficient to only quantify the current state of the material. A forward model of the damage evolution is required to quantify the remaining service life of the component. The evolution of damage within materials is inherently a four-dimensional (4D) phenomenon encompassing all three spatial dimensions plus time. As a result, properly studying the initiation and evolution of damage requires a means of analyzing material structures in three dimensions (3D) at discrete points in time. The time-dependence requires that the analysis be non-destructive, lest the analysis process affect the behavior of the features of interest.

At a given instance in time, a damage region, such as a crack, will occupy a particular region in space. As time progresses and the material is subjected to its operating loads and environment, it is quite likely that the crack will evolve and occupy a new, larger region in space. For many materials, such as metals, crack growth can be defined by the well-known Paris Law (Paris et al., 1961).

$$\frac{da}{dN} = C\Delta K^m \tag{1.1}$$

The red curve in figure 1.2 can be divided into three regions: the initial non-linear region, the central linear region, and the final non-linear acceleration to fracture. The Paris Law is only valid for the central region, appearing linear in the log-log plot. A material will spend the majority of its service life, however, in the first region. This is also the region where early-stage damage, when the damage is of the same length-scale as the material granularity, occurs. In order to make advances in structural health monitoring and prognosis, it must be possible to study the material on its granularity-scale. Early-stage damage does not follow the idealized behavior of equation 1.1, and proper characterization requires direct measurement of granule-scale properties.

A limited number of granular properties may be observed with a simple two- dimensional (2D) technique, such as optical or electron microscopy coupled with image processing. In these cases, the desired 3D information may be deduced from a 2D measurement (e.g., determining

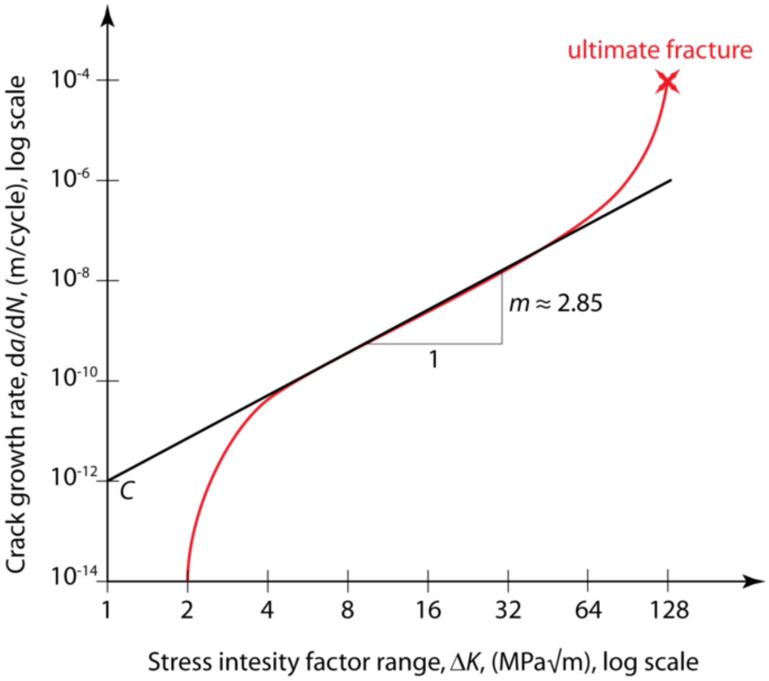


Figure 1.2 Example of Paris Law crack growth.
Image source: http://en.wikipedia.org/wiki/File:ParisLaw.png

volume-fraction from a surface-fraction measurement) but such an approach is often limited to providing simple 3D quantities and requires either simple material structures or simplifying assumptions of more-complex materials. These 2D observations are limited to the outer surfaces of the sample, requiring destructive sectioning to perform a true 3D analysis. Further, determining 3D characteristics from 2D measurements requires a large number of samples and the proper sampling of the statistics involved with the early-stage damage. In many cases the required number of samples is too large to be economically feasible. These limitations render 2D measurement techniques impractical for studying the initiation and evolution of early-stage damage.

A further measurement complication is that it is often desirable to perform these measurements *in situ*, while the sample is under a mechanical and/or thermal load. In some cases this is simply to improve detectability (e.g., an open crack can be easier to detect than a tightly-closed crack), while in other instances the applied load may be critical to properly controlling the damage state (e.g., preventing stress relaxation) (Buffiere et al., 2010). As when studying time-dependent phenomenon, such as damage evolution, destructive methods must be avoided for *in situ* studies as it is impossible to appropriately section the sample and collect measurements as it is under load.

Studying material damage, especially at the early stages, requires very fine spatial resolution, as determined by the granularity length scale, which severely limits the field of allowable measurement techniques. Early stage damage occurs on the granular scale of the material, and the measurement technique must be capable of resolving this scale. An effective technique for directly imaging these small-scale internal structures is x-ray computed tomography (CT), more-specifically the subset known as “high-resolution CT” (HRCT) or “micro-CT” (μCT). This subset of CT focuses on the analysis of small-scale samples and structures with spatial resolutions ranging from sub-micron to a few tens of microns (Salvo et al., 2003, 2010). In addition to providing a direct method of imaging internal 3D structures at the required resolution, CT is non-destructive and thus suitable for use with damage-evolution and *in situ* studies. Further, CT can provide a significant cost and time advantage over 2D imaging and sectioning in cases where destructive methods would be acceptable.

By leveraging the spatial resolution of HRCT it is possible to perform direct damage-quantification measurements on the granular scale within a material. Such measurements are necessary for model input as well as model validation and verification. Coupling the HRCT result with lower-cost measurement results enables the identification of relevant signatures within the low-cost measurement, which is a core requirement if low-cost measurement techniques are to be used for structural health monitoring and prognosis.

Unfortunately, achieving reconstruction voxel sizes on the order of single microns has not been possible in typical laboratory facilities. Achieving such resolutions has required access to a synchrotron facility, which introduces significant time and cost constraints for projects which require the study of materials’ granular structure. Performing reconstrctions with voxel sizes on the order of 2 μm, or smaller, is necessary for imaging the granular structure of fiber-reinforced composites. In addition to achieving smaller voxel sizes, the contrast sensitivity of the reconstruction must also be improved in order to capture the subtle, low-contrast signatures of early-stage damage.

An improved HRCT reconstruction capability, capable of achieving the resolution needed for studying fiber-reinforced composites, will function as a catalyst which enables significant advances in the capability to perform structural health monitoring and prognosis on composite structures. These advances are the result of achieving the necessary resolution using a bench-top laboratory system, thereby allowing materials studies to move out of expensive synchrotron user facilities. The improved resolution will enable the development and evaluation of damage evolution models while comparison against alternative measurement techniques will enable proper interpretation of measurement signals produced by field-deployable techniques.

1.1 Contribution of this Dissertation

As noted in section 1.3, despite significant advances in the ability to use CT to image small-scale damage structures in materials, further improvements are needed to enable the study of early-stage damage. Commercially available bench-top systems lack the flexibility required to image granule-scale structures and the cost and scarcity of synchrotron user facilities severely limits their use in materials studies. This dissertation will introduce a new reconstruction algorithm, discussed in chapter 2, which improves contrast sensitivity, reduces cone-beam artifacts, and processes massive datasets in an efficient manner. When coupled with voxel sizes on the order of 2 μm, the improved algorithm is capable of performing reconstructions which resolve fiber-scale structure within fiber-reinforced composites without requiring additional contrast agents.

CT reconstructions produce datasets which are too large for a human operator to efficiently and consistently analyze. Addressing this challenge involves two complimentary approaches. First, the reconstruction algorithm is structured to allow reconstructing a subset, rather than the entire, volume. This allows the computational effort to be focused on a region of interest. The computational cost of CT has been a longstanding bottleneck, and the ability to avoid unnecessary calculations allows significant reductions in computational time. Coupling this subset-reconstruction capability with GPU computing enables the processing of hundreds of gigabytes of data on a common workstation, substantially reducing the equipment cost associated with the CT reconstruction. Focusing the reconstruction on a limited region of interest also aids in the viewing and analyzing of the reconstruction volume by reducing the quantity of data which must be visualized and processed. Second, physics-based image processing algorithms, discussed in chapter 3, are used to provide robust defect detection and characterization capabilities.

Both the improved imaging and the post-processing capabilities will be demonstrated on several materials of engineering interest in chapters 4 and 5. Further, an experimental protocol which leverages the capabilities of these improvements will be proposed in chapter 6.

1.2 History of Computed Tomography

The use of ionizing radiation to study hidden, internal structures began in 1896 with the discovery of x-rays (Röntgen, 1896). Röntgen's initial demonstration included a radiograph of his wife's hand, thereby directing radiography down the path of becoming a medical diagnosis tool. Industrial applications, starting with weld inspections, occured within one month of Röntgen's paper. Röntgen's discovery of x-rays earned him the first Nobel Price in Physics, awarded in 1901 (Nobelprize.org, 2014a).

While the ability to generate projection radiographs was a significant aid to medical diagnoses, the loss of depth information imposed limitations. These limitations were partially broken in 1932 with the advent of "laminography" (des Plantes, 1932). This technique used coordinated motion of an x-ray source and detector to clearly image objects within a single plane of the sample while blurring out-of-plane structures. Initially, each plane-of-interest would require its own scan. However, in 1969 a technique was developed which allowed a laminography-like visualization of *any* plane within the object by superimposing a finite collection of radiographs (Garrison et al., 1969). Visualization of a volume through the combination of discrete radiographs, as compared to the continuous-motion of laminography, is known as "tomosynthesis".

An alternative means of determining the internal structure of an object was demonstrated by Oldendorf (1961). Oldendorf placed two nails, one steel and one aluminum, within a ring of steel nails. He then translated the assembly through a pencil-thin radiation beam to generate a line-trace of photon counts. When only translation was used it was impossible to identify the presence of the two internal nails, let alone the difference in their materials. However, by rotating the sample as it was translated, it became possible to not only discern the presence of the internal nails, but to also observe their differing compositions. The combination of rotation and translation of the sample is similar to the mechanics of the first-generation CT scanners.

CT, as it is known today, was first demonstrated by the British electrical engineer Godfrey Hounsfield while working for EMI (Hounsfield, 1973a). Although the first to couple the mathematical theory with a practical implementation, the requisite mathematics had been independently developed in South Africa (Cormack, 1963, 1964) and Russia (Korenblyum et al., 1958). In each case, the mathematics are closely related to the Radon Transform (Radon, 1917), however Cormack and Korenblyum performed their derivations without knowlege of Radon's work. Hounsfield's patent application (Hounsfield, 1973b) for the first CT scanner referenced Cormack's mathematical work (Cormack, 1963), but dismissed it as being impractical for real-world use. Hounsfield and Cormack would share the 1979 Nobel Prize in Physiology or Medicine (Nobelprize.org, 2014b) for their roles in developing the CT scanner.

1.3 Computed Tomography in Material Studies

While the initial CT scanners provided a significant advance in radiological imaging, they were not able to achieve the spatial resolution required for proper materials studies. Such resolutions were first-achieved in 1981 with the advent of high-resolution CT (HRCT) (Sato et al., 1981) using a tightly-collimated pencil beam of radiation requiring both translation and rotation of the sample during the inspection. The required motion for this proof-of-concept was very similar to that used by Oldendorf (1961) and Hounsfield (1973a), and was not practical for mainstream use. With the advent of more-practical inspection geometries the technique began to see significant use in the 1990s, beginning with the study of geologic materials (Carlson and Denison, 1992).

Although the initial geologic study was done using a standard tube radiation source, many further studies, starting with Kinney et al. (1993), often used synchrotron radiation in order to further-push the limits of detectability. By the late 1990's it was possible to achieve submicron resolution by using a technique known as "phase-contrast imaging" (Snigirev et al., 1995; Cloetens et al., 1996, 1997; Spanne et al., 1999).

Leveraging phase-contrast imaging allowed material scientists to study small-scale deformations and failures, such as detailed analysis of crack-tip growth (Guvenilir et al., 1997, 1999) and fiber-reinforced composites (Maire et al., 2001; Moffat et al., 2010). Naturally, with the ability to image such features came the desire to couple the results with predictive finite element models (FEM) (Maire et al., 2003; Youssef et al., 2005). Utilizing the rapid data-acquisition capabilities of synchrotrons further-improved *in situ* studies by reducing stress-relaxation effects when quantifying crack growth (Toda et al., 2011). In addition to studying materials *in situ* with mechanical loads, such as the above studies of crack-tip growth and fiber failure, CT was used to study *in situ* thermal loads (Limodin et al., 2007, 2009) and solidification behavior (Terzi et al., 2009).

As synchrotron and detector capabilities improved, so did the achievable resolution the reconstruction volumes. Once sub-micron could be imaged, it became possible to dramatically improve the study damage initiation and early evolution mechanisms (Weck et al., 2008; Maire et al., 2008) and to follow the damage through its development (Maire et al., 2012).

Although synchrotrons have been capable of achieving the necessary resolution for studying early-stage damage initiation and evolution, there has been a continual effort to perform similar studies using commonly-available laboratory equipment (Kini, 1994; Fan, 2001; Zhang, 2003; Sheikh, 2006). Synchrotrons are a very expensive, limited resource and a researcher can rarely obtain access for more than a couple weeks per year. Laboratory CT systems, by contrast, are common at many academic institutions and industrial facilities, thus offering greater accessibility for a much lower cost.

Significant improvements in the ability to image material damage and the granular structure of fiber-reinforced composites has been made the past decade, however achieving the necessary resolution to image features on the length-scale of individual fibers has remained a challenge. Previous studies have dealt with this resolution limitation by considering larger-scale structures, such as fiber bundles (Bayraktar et al., 2006; Schell et al., 2006; Djukic et al., 2009a,b), using radio-opaque contrasting agents (Schilling et al., 2005; Tan et al., 2011), or applying intensive post-processing algorithms to the reconstruction volumes (Liotier et al., 2010).

Considering larger-scale structures is not appropriate when studying the initiation and early-stage evolution of damage. The features of interest in this case are on the same order as the individual fibers rather than the fiber bundles or plies.

Additionally, although the use of a contrasting agent is very effective method of imaging small-scale structures it has two significant drawbacks. First, the damage structure must be interconnected so that the contrast agent can fill the structure. And second, introducing a contrast material will affect the mechanical behavior of the sample under further loading, thus making contrast agents unsuitable for studying the evolution of damage structures.

The application of image processing routines to CT reconstruction volumes is complimentary to the above techniques. However, the design and application of image processing routines must be chosen with care in order to avoid introducing artifacts and false-signatures into the result.

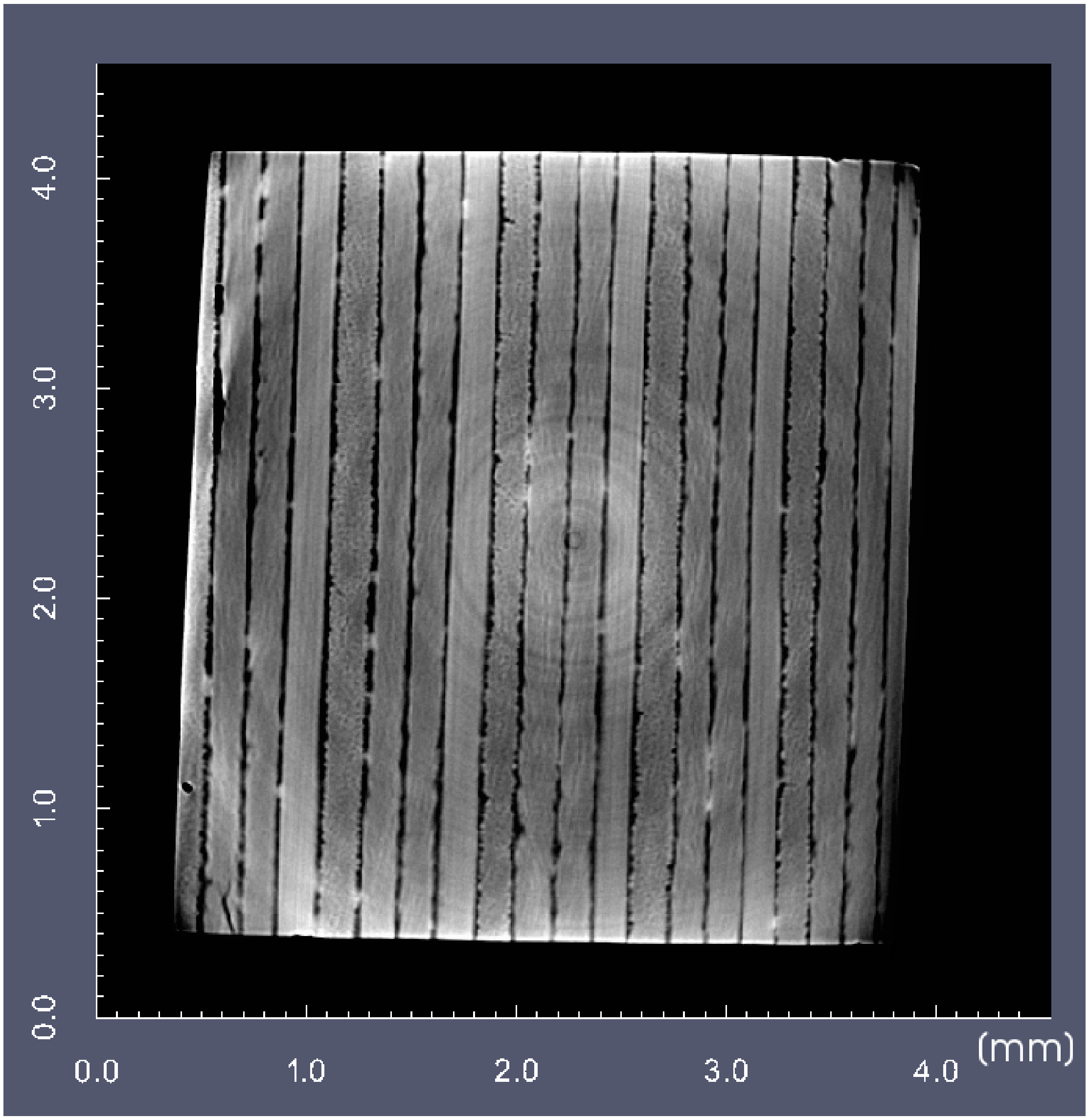


Figure 1.3 Example CT reconstruction slice demonstrating granule-scale resolution in a carbon fiber reinforced polymer composite. The sample has a quasi-isotropic $[0/+45/-45/90]_s$ layup, with the 0-degree fibers coming out of the page clearly visible. The light-colored plies to the left of the 0-degree plies contain fibers oriented at 90-degrees and the ±45-degree plies are to the right of the 0-degree plies. Notice the fiber-scale detail which is visible, particularly in the 0-degree plies. The faint, concentric rings are an artifact caused by the x-ray detector used in the CT scan.

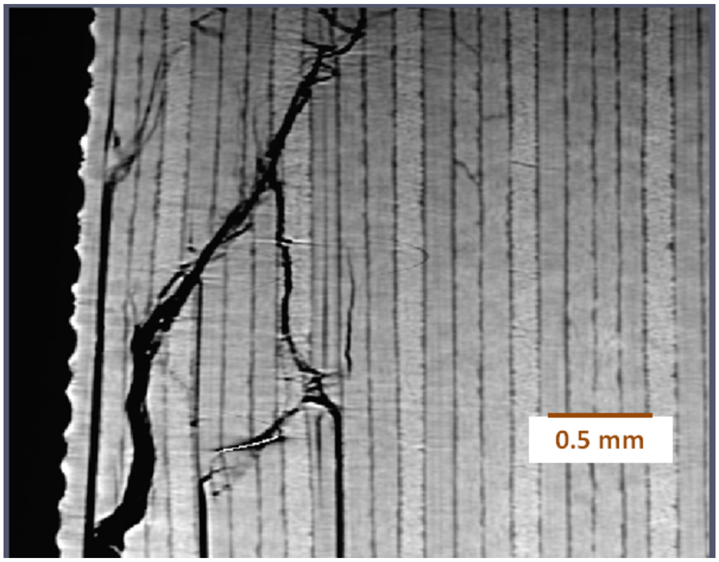


Figure 1.4 Example of CT reconstruction showing complex damage network within carbon fiber reinforced composite. While the inter-ply delaminations are readily seen, and are to be expected, there are also several instances of intra-ply cracks as well as micro-cracks which cut across a single ply.

CHAPTER 2. COMPUTED TOMOGRAPHY RECONSTRUCTION

Performing high-resolution CT (HRCT) on complex materials and imaging granule-scale structures requires careful attention be given to the CT reconstruction algorithm. As stated in section 1.3, historical usage of CT within materials studies has often involved synchrotron radiation rather than laboratory tubes and the use of laboratory tube sources introduces several complications into the reconstruction and analysis processes.

The past decade has seen significant improvements in the ability for laboratory CT systems to image the granular structure of fiber reinforced composites, although achieving the required resolution for studying early-stage damage has remained elusive. This chapter reviews the standard approach to performing filtered back-projection CT reconstruction and discusses novel extensions which significantly improves the quality of the reconstruction. This improved reconstruction quality is obtained by reducing the size of the reconstruction voxels to be on the order of 2 μm and improving the contrast sensitivity in order to detect subtle, low-contrast signatures of early-stage damage. These improvements are necessary for HRCT to become a core tool which is used for performing structural health monitoring and developing prognosis models.

2.1 Reconstruction Theory

Before discussing the new developments in cone-beam CT, consider the theoretical basis for transmission CT. Much of sections 2.1.1 - 2.1.2 is adapted from chapters 7 and 9 of Barrett and Swindell (1981).

2.1.1 Radon Transform

Begin by considering the sample being inspected as a three-dimensional distribution of linear x-ray attenuation coefficients, μ. The distribution of values for μ is unknown and it is this distribution which iks to be obtained as the result of the CT inspection. Mathematically, the goal is to solve for $\mu(\mathbf{r})$, where $\mathbf{r}$ is the position vector within the sample.

If the simpler case where the sample to be a two-dimensional plane is considered, each acquired projection is a line on the detector. A three-dimensional sample will produce a two-dimensional image on the detector. As the photons pass through the sample, they are absorbed along the transmission path, l, according to Beer's Law

$$I = I_0 e^{-\mu l} \tag{2.1}$$

When the sample is non-homogeneous, equation 2.1 must be integrated along the path.

$$I = I_0 e^{-\int \mu(l) dl} \tag{2.2}$$

When considering the two-dimensional sample and its corresponding one-dimensional projection with a parallel beam of x-ray photons, the following coordinate transformation can be performed for any (x, y) point in the sample. Quantities with a subscript "r" are in the rotated coordinate system, and the angle ϕ is the angle which separates the rotated and unrotated coordinate systems. The angle θ is the polar-coordinate for the (x, y) point in the unrotated coordinate system. This geometric setup is shown in figure 2.1.

$$x = r \times cos(\theta) \tag{2.3}$$

$$y = r \times sin(\theta) \tag{2.4}$$

$$x_r = r_r \times cos(\theta_r) = x \times cos(\phi) + y \times sin(\phi) \tag{2.5}$$

$$y_r = r_r \times sin(\theta_r) = -x \times sin(\phi) + y \times cos(\phi) \tag{2.6}$$

$$r_r = r \tag{2.7}$$

$$\theta_r = \theta - \phi \tag{2.8}$$

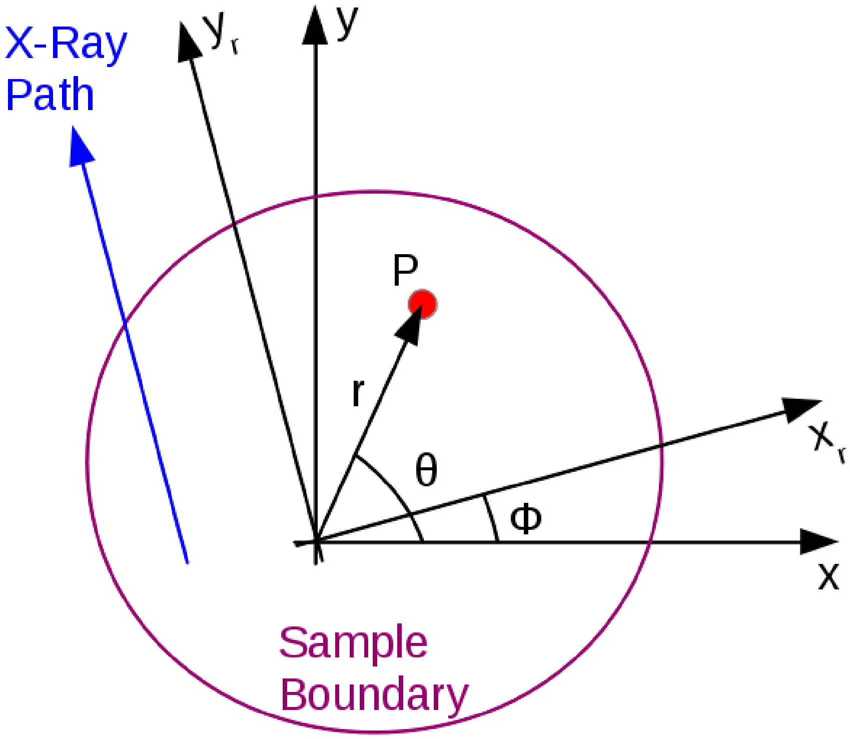


Figure 2.1 Coordinate system transformation for parallel-beam geometry. The x-ray photons travel parallel to the y_r axis. A position, P, located at (x, y) within the sample can be expressed in polar coordinates with coordinates (r, θ). The rotated coordinate system is offset from the stationary system by angle ϕ.

We an express the line integral of attenuation through the sample as

$$\lambda_\phi(x_r) = -ln\left(\frac{I_\phi(x_r)}{I_0}\right) = \int_l \mu(x_r, y_r)dy_r \tag{2.9}$$

Where the subscript ϕ denotes quantities at a specific rotation, ϕ.

Equation 2.9 is the projection of $\mu(x, y)$ onto the detector and is known as the Radon transform of μ. Performing a CT reconstruction requires inverting this equation to solve for $\mu(x, y)$.

2.1.2 Back-Projection

Back-projection is the process of taking the $\lambda_\phi(x_r)$ projection values of equation 2.9 and “smearing” them backward along the original ray-paths. This operation is repeated for all values of ϕ, and each intermediate result is summed. The summation image is known as the back-projection result. Mathematically, for a parallel-beam geometry,

$$b(x,y) = \frac{1}{\pi}\int_0^{\pi} \lambda_{\phi}(x_r)d\phi \tag{2.10}$$

with

$$x_r = r \times cos(\theta - \phi) \tag{2.11}$$

where (r,θ) describes the point (x,y) in the sample in terms of the Radon transform and ϕ is the rotation angle for the projection, the point-spread-function (PSF) of the back-projection operation can be calculated using the Dirac delta, δ, as a point-source

$$p(r) = \frac{1}{\pi}\int_0^{\pi} \delta(r \times cos(\theta - \phi))d\phi \tag{2.12}$$

which has the solution

$$p(r) = \frac{1}{\pi r} \tag{2.13}$$

The back-projection result, $b(r)$, can be expressed as a convolution of the true attenuation distribution with the PSF

$$b(r) = \mu(r) \circledast p(r) = \mu(r) \circledast \frac{1}{\pi r} \tag{2.14}$$

where $\circledast$ indicates convolution.

Notice that the PSF in equation 2.13 does not depend on the (x,y) position of the point-source within the sample, but rather only on the distance from the point-source. This position-invariance occurs with both parallel-beam and fan-beam inspection geometries. The PSF introduces a blurring artifact into the back-projection reconstruction result which may be compensated through the use of an appropriate deconvolution operation.

2.2 Inspection Geometry

The PSF of the CT reconstruction process is a function of the inspection geometry. As such, consider the inspection geometry used at CNDE before discussing the details of the PSF-compensation filter in section 2.4.

The CT inspection systems at CNDE use a stationary source and flat-panel detector coupled with a four-axis (three-dimensional translation plus rotation) sample positioner motion stage. A photograph of the interior of the high-resolution CT system is shown in figure 2.2.

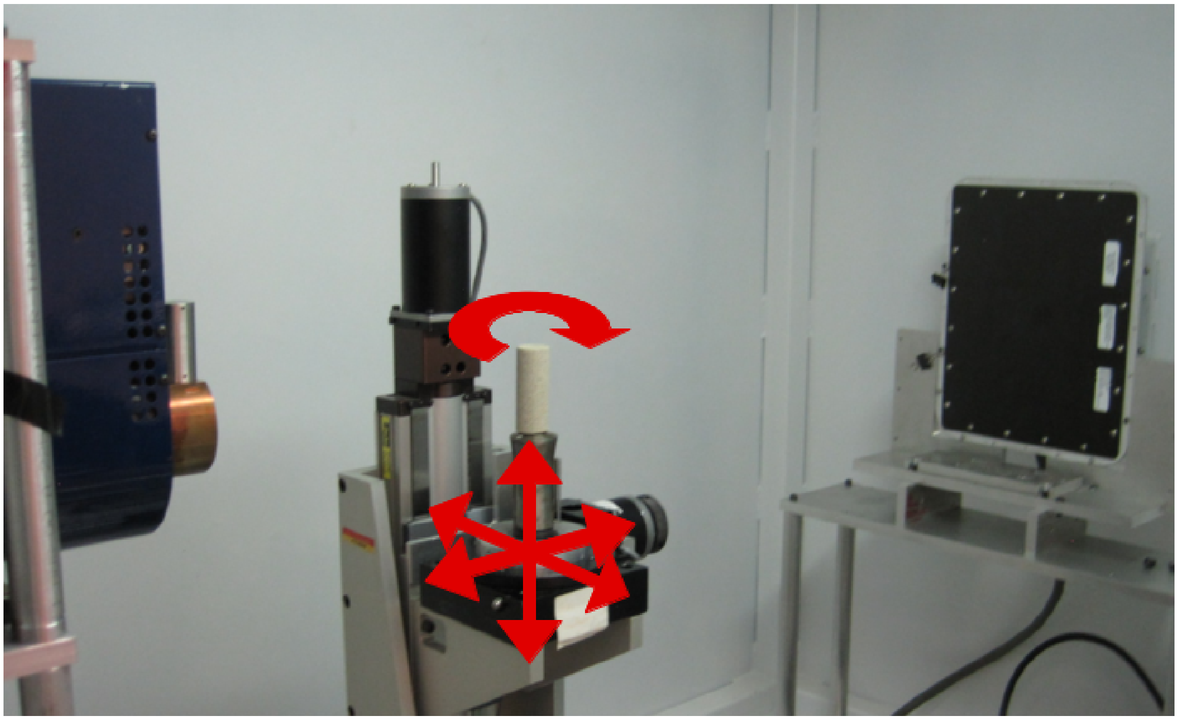


Figure 2.2 Interior of high-resolution CT vault. The sample can be translated in all three spatial dimensions, as well as rotated about the vertical axis. The microfocus x-ray tube source is on the left and the flat-panel x-ray detector is on the right.

A microfocus x-ray source is used and has a focal spot size of 20 μm or less, depending on the tube power settings. The focal spot is the region within the tube from where the x-rays are produced and smaller spot sizes produce sharper images and allow for greater geometric magnification.

Geometric magnification is controlled by the position of the sample between the source and detector. The x-rays generated by the tube form a cone of radiation which travels to the flat-panel detector. The divergence of the rays as they travel from the source to the detector cause the projected image of the sample to be magnified, and the magnification is a function of the sample's position.

$$\text{magnification} = \frac{\text{source to detector distance}}{\text{source to sample distance}}$$

The sample is positioned such that its projection onto the x-ray detector fills as much of the detector as possible. This is required to maximize the spatial resolution of the scan. The scan itself is performed by acquiring a series of projection images of the sample with a small, typically 1 degree, rotation of the sample occurring between each exposure.

It is also possible to perform a CT scan involving sample translations and rotations about other axes between exposures, and when using a cone-beam inspection geometry, as compared to the traditional third-generation data acquisition method. Feldkamp et al. (1984) showed that when the sample is only rotated about a single axis in a cone-beam inspection, there is an incomplete sampling of the three-dimensional frequency space and as a result it is impossible to perfectly reconstruct features which are parallel to the rotation axis.

Experimentally implementing such a system is difficult due to the need to precisely quantify the sample's motion and thus it is typical for CT scans to involve no translation and only rotate the sample about a single axis. A scan including a series of tilts spanning ±20 degrees was attempted for a small sample mounted on a goniometer which allowed the sample to be tilted during the scan. Due to the difficulties with quantifying the sample's post-tilt position to sufficient accuracy, it was not possible to utilize the extra, non-standard exposures in the experimental system at CNDE.

This geometry was simulated using XRSIM, a physics-based forward model developed at CNDE. The effect of including a full rotation about the 2nd axis scan can be seen in figure 2.3. The reconstruction result obtained when only rotating about a single axis is shown on the left while the 2-axis result is on the right. Slight improvements to regions at the far edges of the cone beam are gained by adding the rotation about the second axis, but these improvements are very minor. Such a scan is also exteremely difficult to implement experimentally, and when the negligible difference in reconstruction quality is considered it can be concluded that the conclusion that the standard inspection geometry (rotation about a single axis, with no other rotations or translations during the scan) is sufficient when considering actual, rather than theoretical, application.

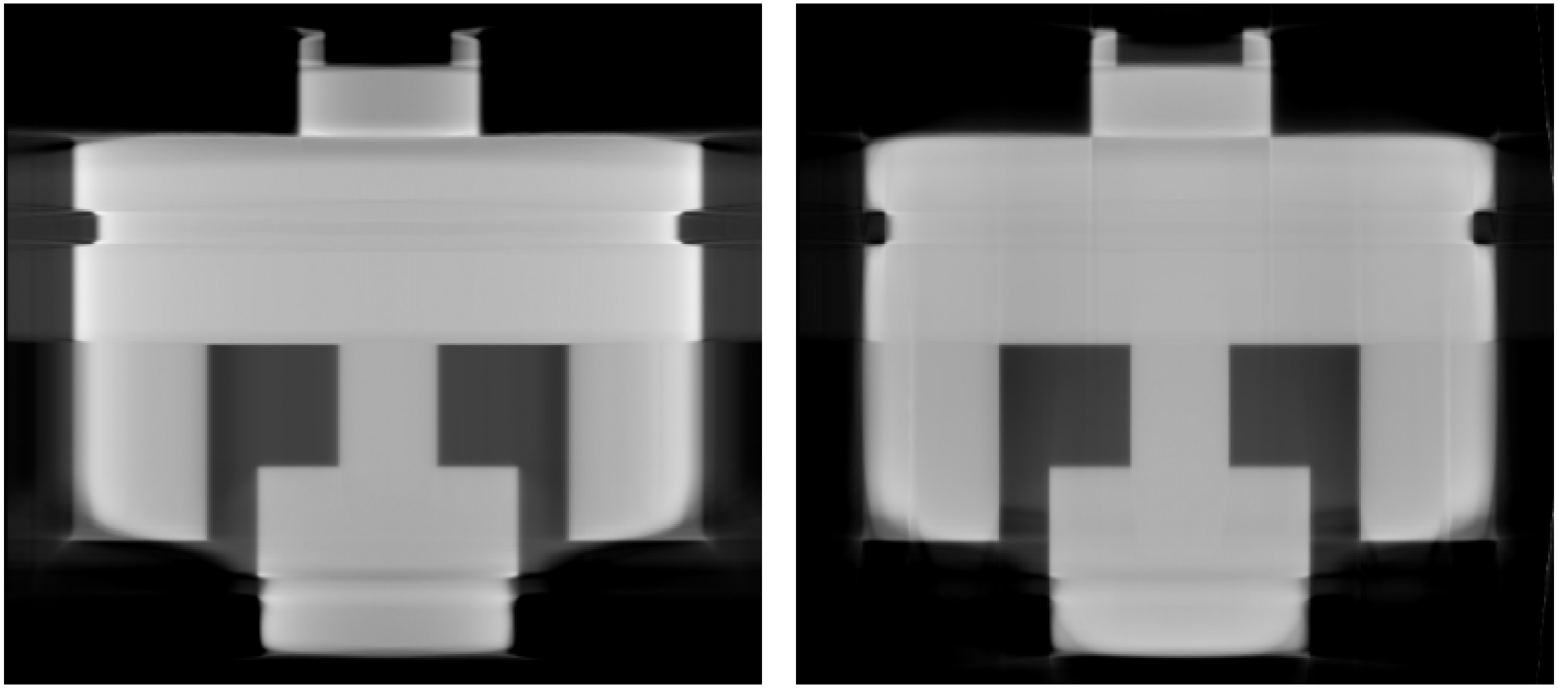


Figure 2.3 CT reconstruction result showing artifacts caused by rotation about a single axis during the scan. The sample on the left was rotated around the vertical axis in the figure and the sample on the right was rotated around both the vertical and horizontal axes in the figure. Slight improvements to regions located near the top and bottom edges of the cone beam are obtained by including the additional rotation axis, but the improvements are very small. Experimentally implementing such a scan is very difficult, and is not justified by the marginal change in reconruction image quality.

2.3 Magnification Effects

Studying the granular structure of complex materials requires the use of small samples at high magnification. When the magnification is pushed to the limits of the system, such as when the sample is as close as physically possible to the tube, additional complications are introduced. At large magnifications, the difference in magnification between the source and detector sides of the sample becomes significant for regions near the top and bottom of the cone-beam, as shown in figures 2.4 and 2.5.

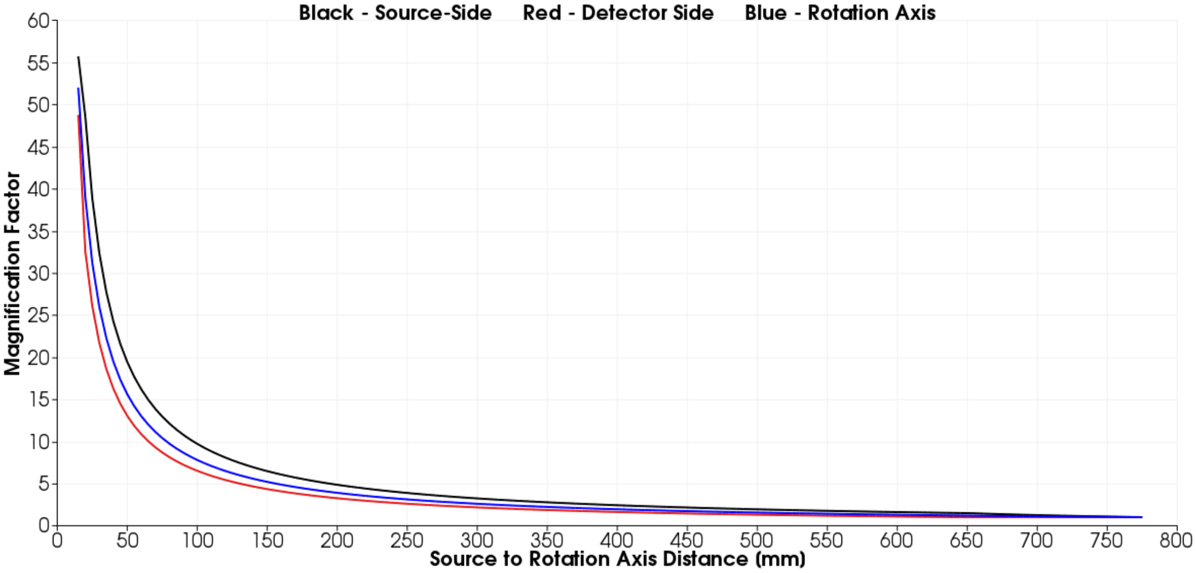


Figure 2.4 Magnification as a function of position.
Blue line is the magnification of the rotation axis and would be considered the nominal magnification of the CT scan.
Red line is the magnification of the detector-side of the sample.
Black line is the magnification of the source-side of the sample.
Source-to-detector distance is 780 millimeters.

This difference in magnification causes the projection of a point within the sample to change position vertically on the detector as the sample is rotated, as shown in figure 2.5. When parallel-beam geometry is used, such as in a synchrotron, there is no vertical component to the motion of a projected feature. As the magnification of the sample increases, so does the difference in magnification between the front and back surfaces. This growing magnification-difference increases the vertical motion of a projected point, and although this vertical motion can be ignored in many instances, (Feldkamp et al., 1984; Bronnikov, 2000) it must be addressed when the vertical filed-of-view grows sufficiently large, such as during a high-magnification inspection.

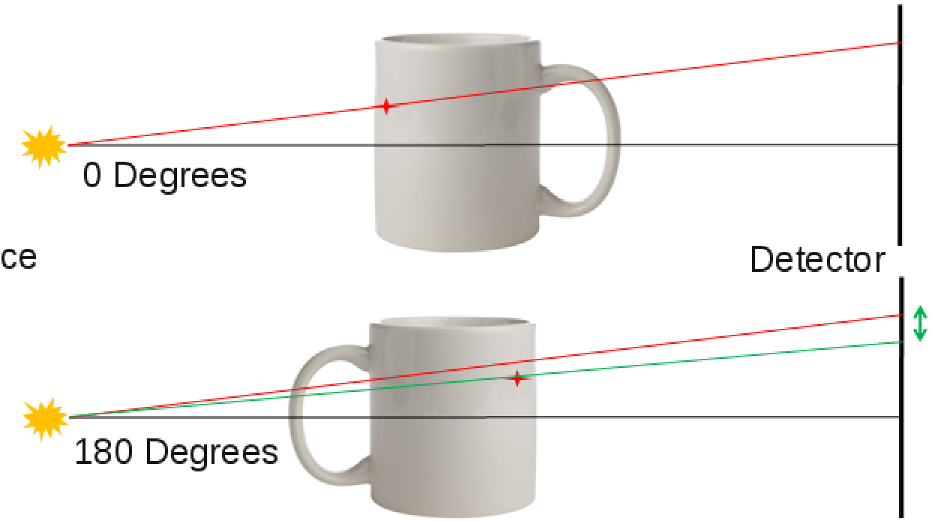


Figure 2.5 Illustration of the significance of the magnification difference between the source and detector sides of a sample at high magnification. Notice how the projection of the selected point, identified by the red star, moves vertically on the detector as the sample rotates. This is shown by the red and green line traces, representing the point's projection, intersecting the detector at different vertical positions.

Another challenge introduced by high magnification concerns the effects of the source's finite spot size. A point-source will produce crisp edges at any level of magnification while a finite source will introduce a penumbra, or shadow, known as "geometric unsharpness", denoted U_g. The size of the shadow is

$$U_g = f\frac{b}{a} = f(\text{magnification} - 1) \tag{2.15}$$

where f is the size of the focal spot, b is the distance from the sample to the detector, and a is the distance from the source to the sample. This is shown schematically in figure 2.6.

As magnification is increased, the ratio of $\frac{b}{a}$ increases, causing the geometric unsharpness to increase as well. This can be mitigated to through the use of a microfocus x-ray source. However, at sufficiently large magnifications the penumbra can span several pixels on the detector even when a microfocus source is utilized. It is common to avoid increasing the magnification beyond the level where the source focal spot becomes larger than the effective detector pixel

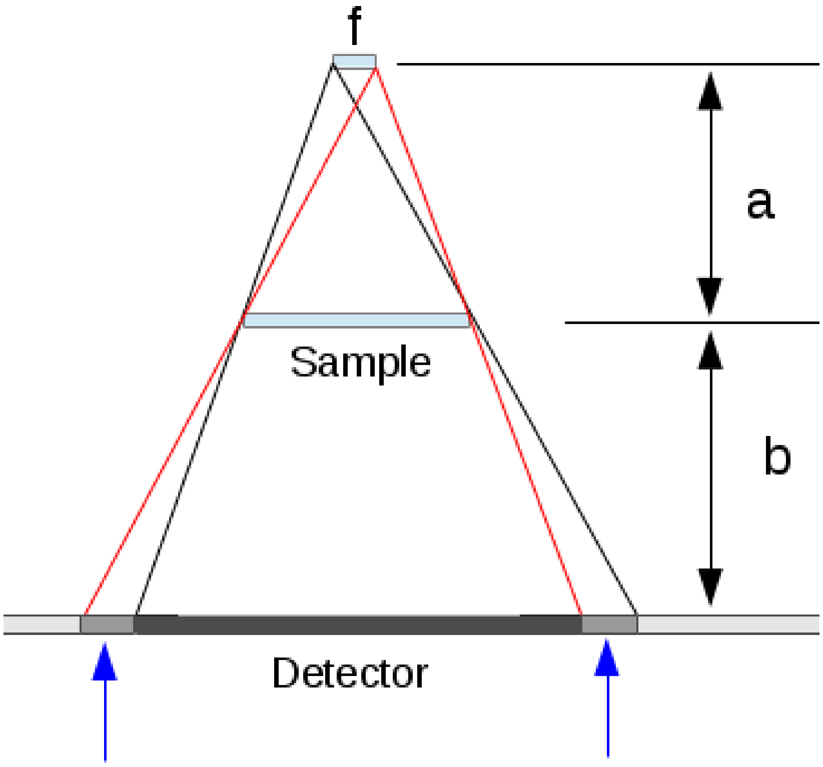


Figure 2.6 Calculation of geometric unsharpness. The penumbra, or shadow, created by the finite focal spot is identified by the blue arrows.

size.

To explore the effect of geometric unsharpness on high-magnification CT scans a spherical glass bead which has a diameter of 660.4 μm was scanned. A slice from the reconstruction can be seen in figure 2.7.

The bead diameter was measured in the reconstruction as 285 voxels, which requires the size of each voxel to be 2.3 μm. This corresponds to a magnification factor of 43.5. When looking at the profile of the edge of the bead, shown in figure 2.8, it can be seen that the transition occurs over approximately 15 voxels, which spans 34.5 μm in the slice. In an idealized scan the transition would be a perfect step.

There are several factors which combine to blur the perfect step into the transition seen in figure 2.8: detector noise, the influence of the decreasing thickness of the sphere as the edge is approached, the smoothing effect of the point-spread-function compensation filter discussed in section 2.4, and geometric unsharpness. All of these factors, except for the sphere geometry,

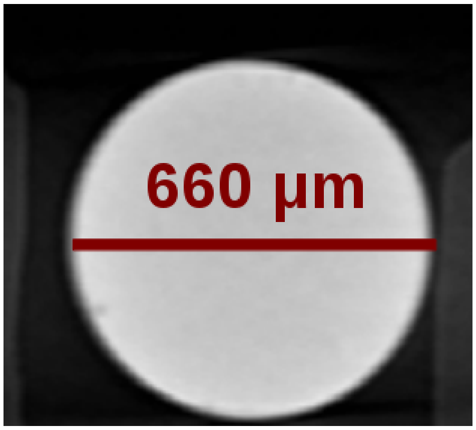


Figure 2.7 CT reconstruction of glass bead. Actual bead diameter is 660.4 μm.

are present for the inner diameter of the plastic cylinder which contains the sphere during the scan. The line trace in figure 2.8, in an idealized scan, would contain a perfect step for the surface of the plastic cylinder. The observed blur in this case is approximately 6 voxels, or 13.8 μm. Since only the sphere geometry is not captured by this edge, it can be determined that the sphere geometry accounts for 9 of the 15 voxels spanning the blurred edge of the sphere. If the entirety of the 6 voxels spanning the cylinder's blurred edge are attributed to geometric unsharpness, the spot size can then be caluculated to be 14.1 μm which agrees with the spot size reported by the control software. In reality, detector noise and the PSF compensation filter do play a role, and the geometric unsharpness is thus caused by a spot which is smaller than the reported size.

It is important to note that the uncertainty manifested as blurred edges primarily affects the ability to perform precise measurements using the CT reconstruction result. It is still possible to detect the presence of features smaller than this unsharpness, although it will not be possible to accurately quantify them. This is demonstrated by the ability to detect the presence of a small bubble in the lower-left portion of the bead shown in figure 2.8, but the inability to accurrately measure its size due to the blurring.

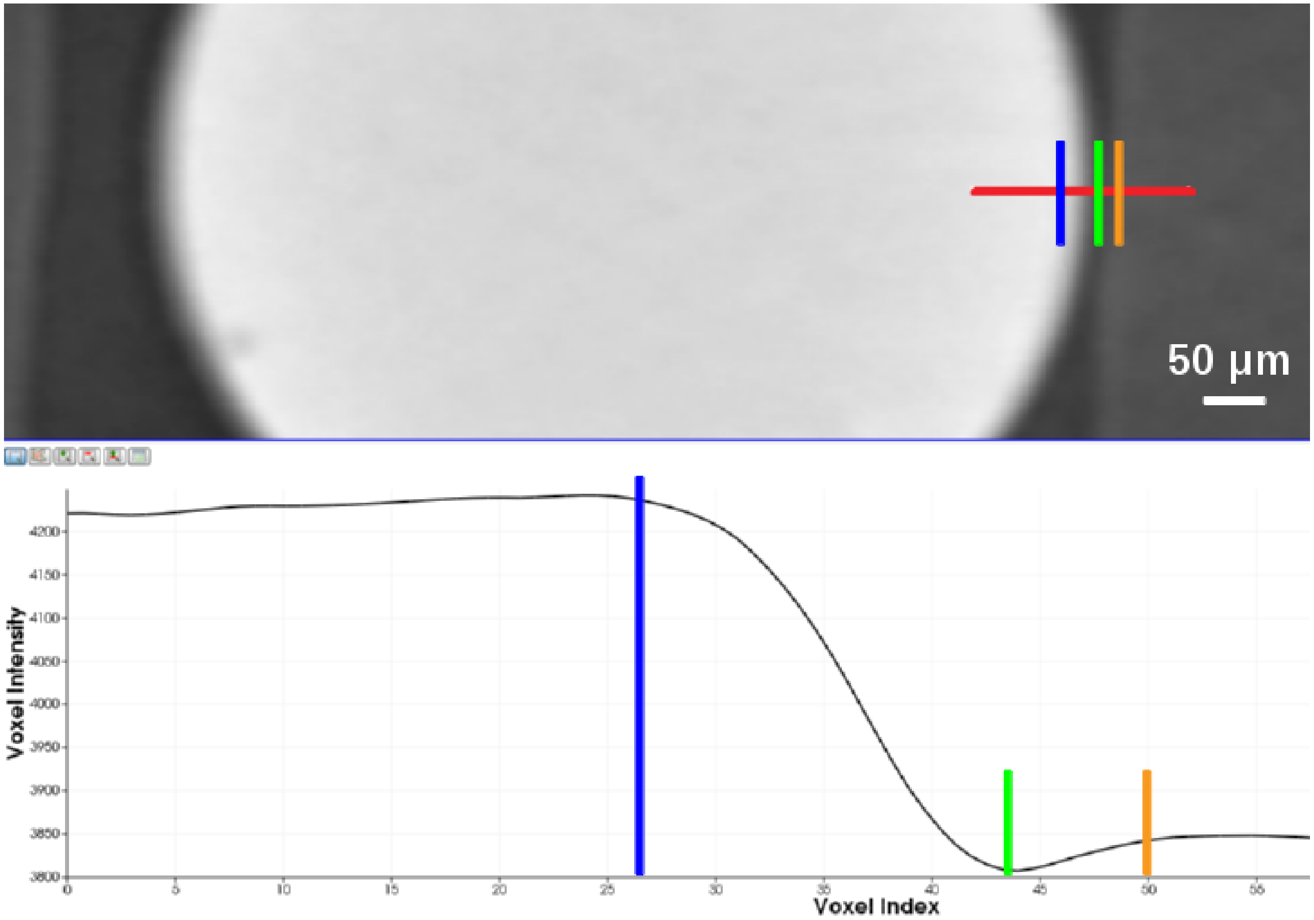


Figure 2.8 Edge profile of glass bead. Actual bead diameter is 660.4 μm. Voxel size is 2.3 μm and the edge transition of the glass sphere spans 15 voxels, or 34.5 μm.

2.4 Back-Projection Point Spread Function

When developing a filter to compensate for the PSF in both parallel-beam and fan-beam geometry, it is mathematically equivalent to apply either a one-dimensional filter to the projection data prior to back-projection or a two-dimensional filter to the back-projection result (Barrett and Swindell (1981)). In the case of cone-beam reconstruction, the filter choices become two-dimensional and three-dimensional, respectively. Conventionally, the lower-dimensional filter is applied prior to back-projection, rather than the higher-dimensional filter applied after back-projection, due to it being faster to calculate and apply the lower-dimensional filter.

In the case of parallel-beam and fan-beam inspections, there is a well-defined position-invariant point-spread-function (PSF) associated with the back-projection operation, as shown in equation 2.13. In general, cone-beam CT does not have a position-invariant PSF. Historically, a fan-beam approximation is used so long as the cone-beam angle is kept small. This corresponds to small values for angle β in figure 2.9. Feldkamp et al. (1984) was the first to introduce this small-angle approximation for cone-beam reconstruction and his algorithm remains in common-use today.

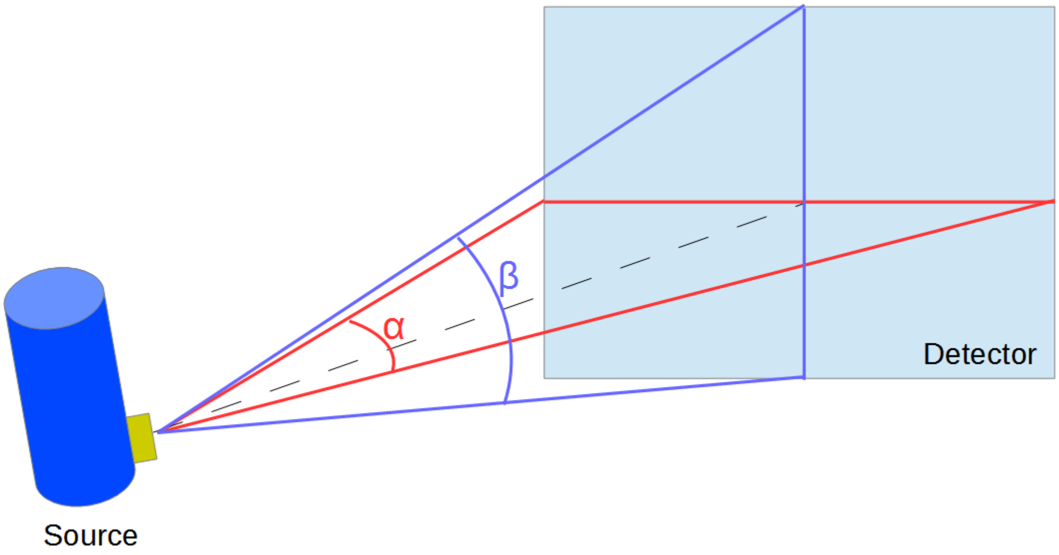


Figure 2.9 Cone-beam angles. The angle α is called the fan-beam angle, and is the angular spread of the beam horizontally along the detector. The angle β is called the cone-beam angle, and is the angular spread of the beam vertically along the detector.

2.4.1 Fan-Beam Inspection Geometry

Barrett and Swindell (1981) shows that the PSF for a fan-beam inspection of a two-dimensional sample has the form

$$p(r) = \frac{1}{\pi r} \tag{2.16}$$

The ideal filter for compensating this PSF is given by Bracewell and Riddle (1967).

$$Q(\omega) = \pi\omega \tag{2.17}$$

This filter was first used by Ramachandran and Lakshminarayanan (1971) and modified by Shepp and Logan (1974) to lightly-attenuate the high frequencies in the interest of noise suppression. Outside of qualitative comments regarding noise suppression, the CT literature does not discuss the effects of modifying the filter from the ideal PSF-compensation form. Section 2.4.2 will address this issue.

When imaging three-dimensional samples with two-dimensional projection images the PSF becomes

$$p(r) = \frac{1}{2r^2} \tag{2.18}$$

Taking the Fourier transform of 2.18, using σ as the two-dimensional frequency vector within the image plane,

$$P(\sigma) = \frac{1}{2\sigma} \tag{2.19}$$

Equation 2.19 indicates that the ideal filter is

$$Q(\sigma) = 2\sigma \tag{2.20}$$

so that $P \times Q = 1$.

Equation 2.20 is a simple linear-ramp in the frequency domain. Unfortunately, a high-pass filter such as this will serve to amplify noise, so in-practice the ideal filter shown in equation

2.20 is combined with a window function to attenuate the noise-amplification. The choice of windowing function plays a significant role in the quality of the reconstruction. A common choice is a Hamming window, though it introduces a lobed structure into the final filter form as seen in figure 2.11.

The Hamming window function has a tendency to introduce spatially-correlated noise into reconstructions of low-contrast materials. This spatial correlation obscures small features and hinders the ability of HRCT to image the subtle, low-contrast signatures associated with early-stage damage.

One would expect the random noise to be uncorrelated and thus vary on a pixel-to-pixel basis, however, it is observed that when imaging low-contrast materials the dominant noise variation occurs over a length-scale of several pixels. This artifact is easily seen in figure 2.10.

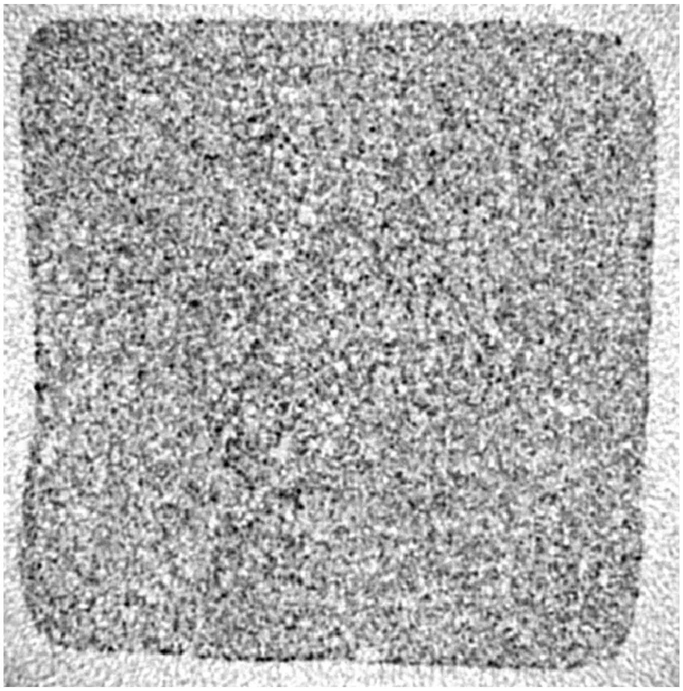


Figure 2.10 Cross-section of low-density foam, reconstructed using a Hamming window with the PSF filter. Notice how the spatial correlation of the noise makes it impossible to discern the internal structure of the foam.

2.4.2 Improving Filter Performance

The quality of the CT reconstruction can be improved if an alternate windowing function is chosen. This effect is particularly noticeable when imaging low-contrast materials such as low-density foam or carbon-fiber composites.

By selecting an exponential-decay window function in the frequency domain, rather than the typical raised-cosine Hamming window, it is possible to achieve significantly-improved image quality in the reconstruction. This window has the form

$$W(\sigma) = e^{-a|\sigma|} \tag{2.21}$$

It is important to note that although this window function has been found to work well for HRCT imaging of low-contrast features, it has not been optimized. This window function was chosen for the ability to modify its behavior, if necessary, for particularly-troublesome datasets and its computational efficiency on graphical processing units (GPUs).

Varying the coefficient a allows for the filter behavior to be easily modified as-necessary for particular datasets. As a tends towards 0 the window has a reduced effect and the PSF compensation filter approaches the mathematically-ideal form, and greater values of a will more-aggressively suppress noise at the expense of introducing blurring. Typically an acceptable trade-off is found for $a = 2$.

This window function is combined with the ideal linear-ramp (equation 2.20 to produce the improved PSF-compensation filter

$$Q(\sigma) = 2\sigma e^{-a|\sigma|} \tag{2.22}$$

A selection of filters produced by this window function can be seen in figure 2.11. For comparision, the filter produced using the Hamming window is labeled “Hamming Window”.

Using the improved PSF-compensation filter produces a superior reconstruction of low-contrast materials, as seen in figure 2.12. Figures 2.10 and 2.12 image the same slice in the same sample, and thus makes the superiority of the improved PSF-correction filter trivial to see.

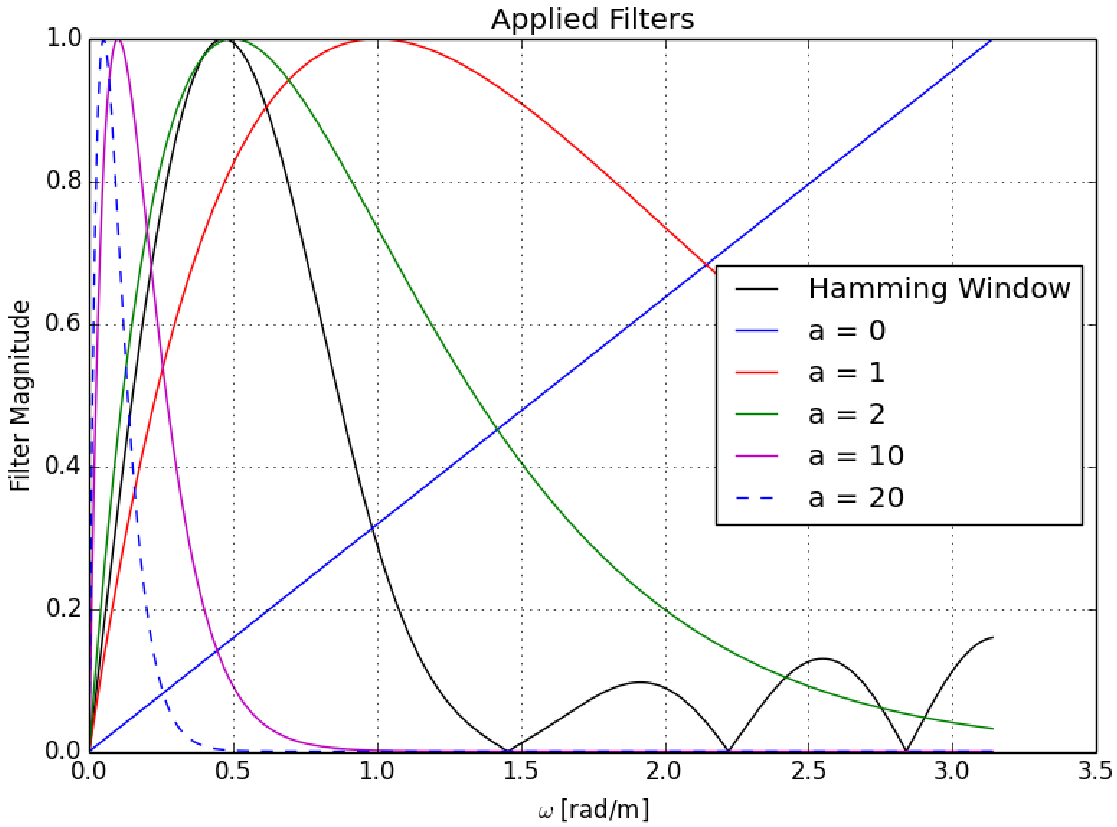


Figure 2.11 Frequency-response of PSF-correction filter for varying values of a. Note the smooth shape for all filters except the one labeled "Hamming". Additionally, note that the filter peak for "Hamming Window" and $a = 2$ very-nearly correspond to the same frequency. Filter maximum amplitudes are scaled to unity to improve their display.

The spatial correlation of the noise in figure 2.10 is visible in regions which should consist of uniform values such as the free-space surrounding the sample.

2.4.3 Validation using XRSIM

As stated above, $a = 2$ provides an acceptable balance between noise-suppression and image clarity. Identifying an appropriate value for a made extensive use of XRSIM, a physics-based forward model developed at CNDE for simulating x-ray inspections. Use of simulation software allows complete knowledge of all input parameters and enables the user to compare the results of processing algorithms to the known inputs.

Figure 2.13 shows the effect of varying a when reconstructing a simulated dataset of an

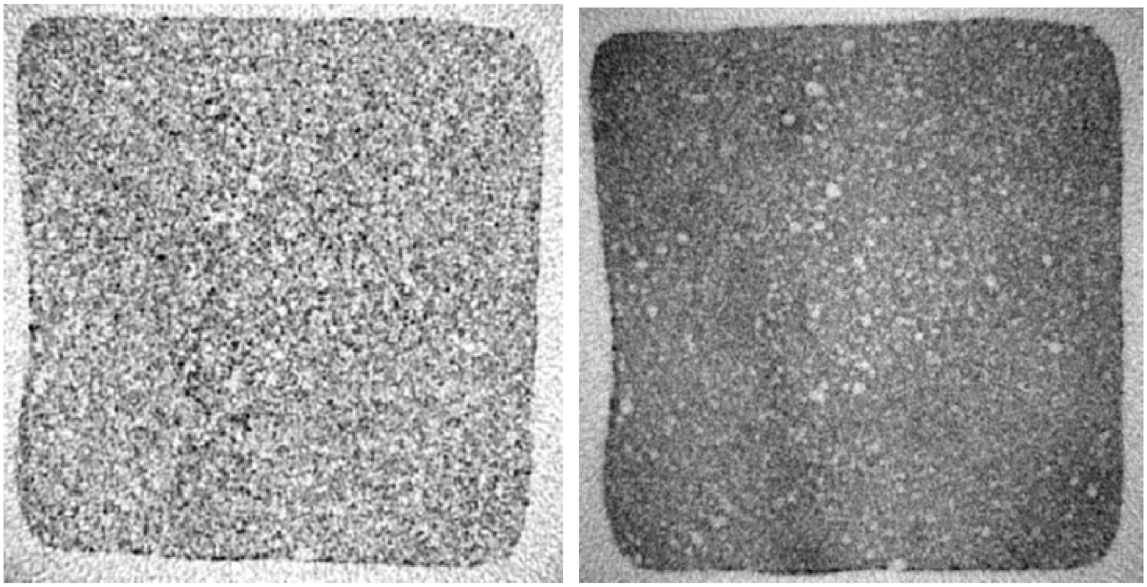
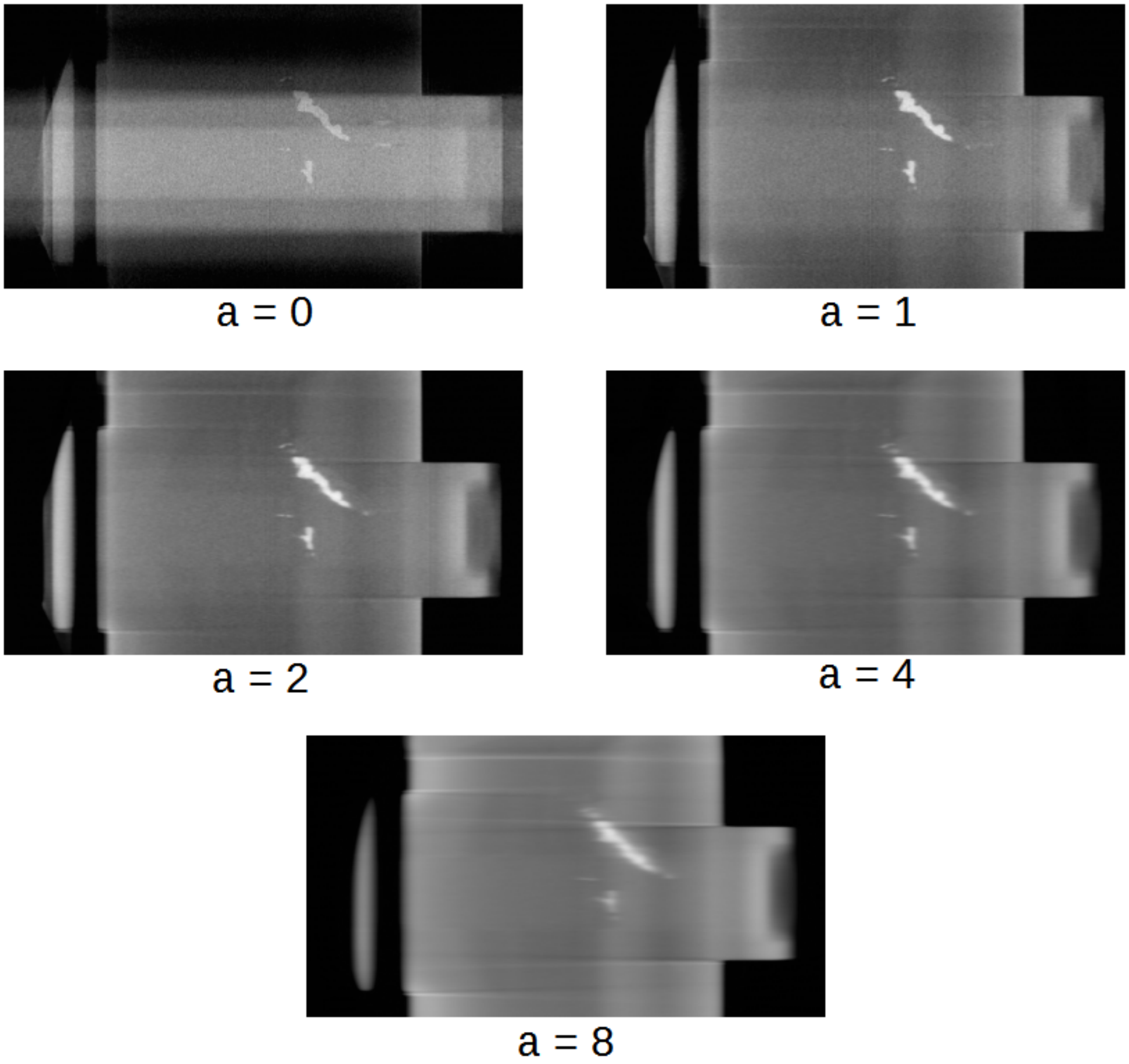


Figure 2.12 Increased image quality due to use of improved PSF filter. The Hamming window was used on the left image and the improved filter was used on the right. Notice the lack of spatial correlation of the noise and how it is now possible to discern the internal structure of the foam.

aluminum casting with a high-density inclusion. We can see in figure 2.13 that the theoretically-ideal filter, labeled ”a = 0” in the upper-left corner of the image, produces sharp edges at the boundary of the inclusion, but the magnitude of the background noise makes it difficult to discern the smallest inclusions. Additionally, the varying thickness of the part introduce intensity variations which are amplified by the theoretically-ideal filter.

As a is increased, the background noise is smoothed and the intensity variations are avoided. When $a = 1$ and $a = 2$, the high-density inclusion retains its sharp edges and the increasing suppression of noise aids imaging of the smallest inclusions. When a is increased beyond 2, however, there is minimal additional suppression of noise and the inclusion becomes progressively blurrier as a is increased. Thus, $a = 2$ provides the best balance between noise-suppression and feature clarity. The reconstruction slices shown in figure 2.13 are representative of behavior observed across a wide variety of samples, both in simulated data and experimental data.



a = 0

a = 1

a = 2

a = 4

a = 8

Figure 2.13 Effect of varying coefficient a in the improved window function. CT data was generated using XRSIM. Note that $a = 2$ provides the best compromise between noise-suppression and image clarity. Also note that $a = 2$ allows the smallest satellite inclusions to be shown.

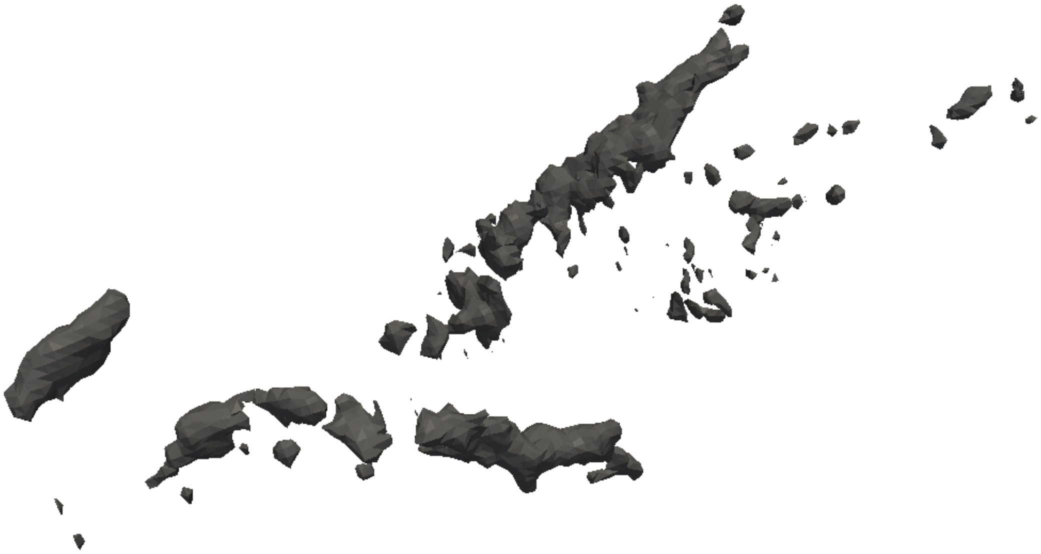


Figure 2.14 CAD rendering of inclusion imaged in figure 2.13. Note the presence of several small inclusions and the complex geometry. Use of a complex-geometry sample in the simulation demonstrates the filter performance on realistic, non-idealized data.

2.5 Cone-Beam Back-Projection

Having addressed the filtering requirements for cone-beam back-projection, now consider the back-projection operation itself. The following section describes the implementation of the cone-beam back-projection algorithm currently used at CNDE. The filters described above are applied to the projection exposures prior to performing these back-projection calculations.

The geometric setup for a typical CT scan is shown in figure 2.15. In most cases the sample is not translated between exposures and the source is centered on the detector. Such constraints are not required, however, and the back-projection process described in this section is readily generalized to include any motion and alignment of the sample.

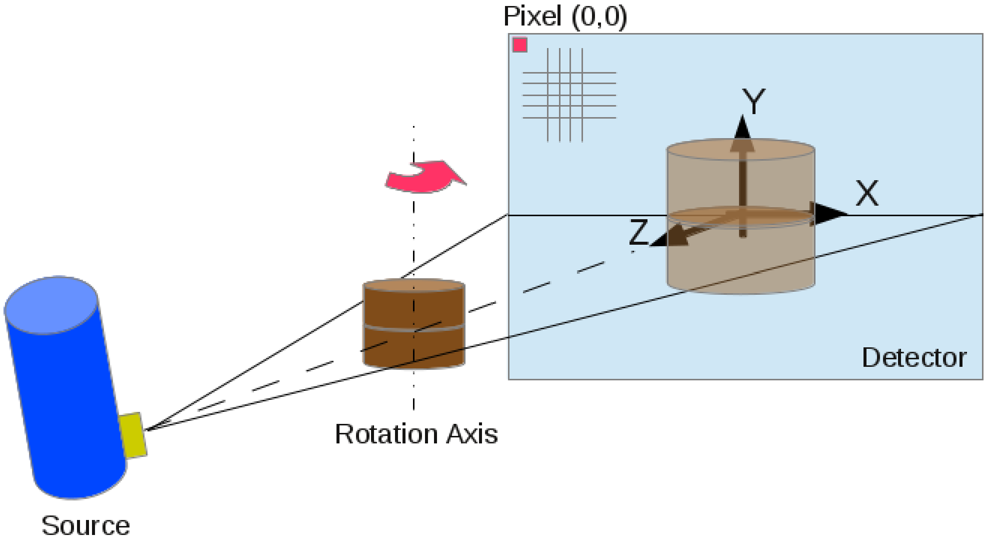


Figure 2.15 Typical CT scan geometry. Sample does not translate between exposures, and uniformly increments rotation about vertical rotation axis. Sample is represented by the dark brown cylinder, and the projection of the sample is represented by the translucent brown cylinder on the detector plane. The global system coordinate axes are fixed to the detector center, with the X-axis parallel to detector rows, Y-axis parallel to detector columns, and Z-axis perpendicular to the detector plane. X-Ray source is defined to be centered at the detector.

2.5.1 Calculating Ray-Paths

A side-view of the geometry in figure 2.15 is shown in figure 2.16. It is easily seen from the side-view that a right triangle is formed by the beam centerline, detector, and ray which passes through the center of the voxel-of-interest. The triangle is drawn explictly in figure 2.17 which has removed the non-essential items in order to more-clearly show the projection geometry.

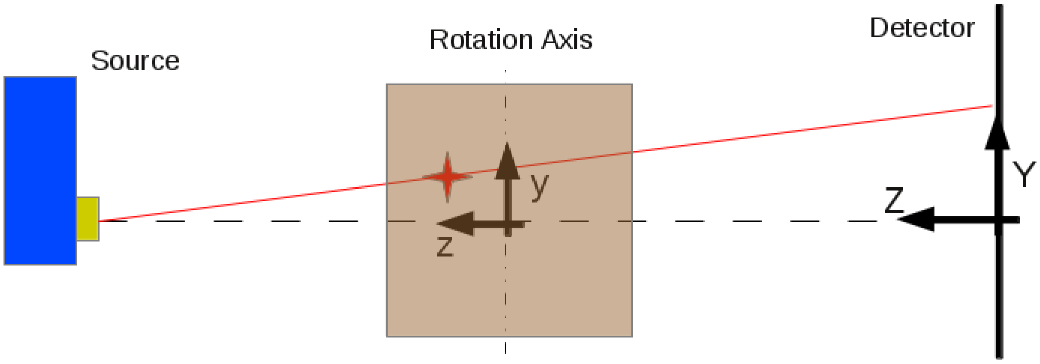


Figure 2.16 Side view of typical CT scan geometry. Voxel-of-interest is identified with the red star, and the ray which passes through the voxel is identified with the red line. The global coordinate axes are located at the detector center, and a sample local coordinate axes are shown centered on the sample.

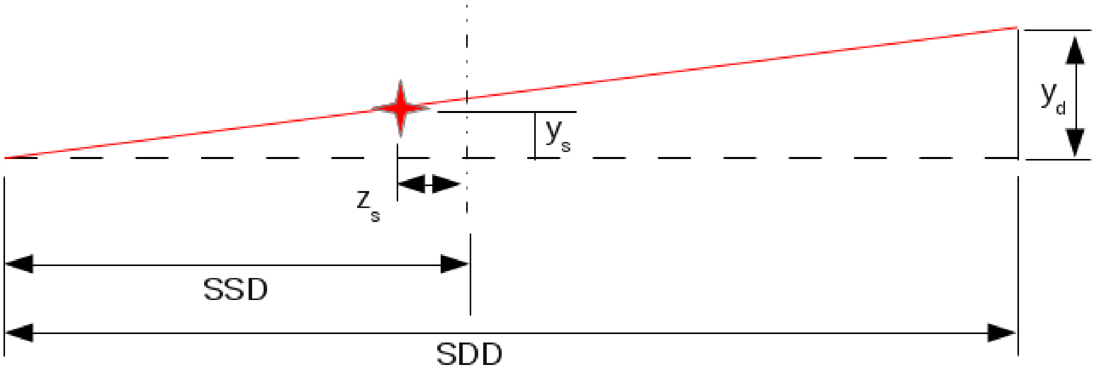


Figure 2.17 Projection triangle formed by CT scan geometry. Solving for y_d, the voxel position projected onto the detector, provides the information required for the back--projection operation. "SSD" is the source-to-sample distance, "SDD" is the source-to-detector distance, and (y_s, z_s) is the voxel position within the sample.

This projection-mapping is the cone-beam version of the integrand of equation 2.9. Similarly, the detector value located at the intersection of the ray and the detector, identified by coordinates (x_d, y_d), is analogous to λ_ϕ in equation 2.9. Further, the integral of λ_ϕ over all angles ϕ produces the back-projection summation image, similar to $b(x, y)$ in equation 2.10.

In general, the position $(x_d,\, y_d)$ will not coincide with a detector pixel-center, and thus the corresponding detector value must be estimated using some form of interpolation. Nearestneighbor can often produce acceptable results, however bilinear interpolation will provide a smoother reconstruction result.

Calculating the ray-detector intersection (x_d, y_d) is easily performed using similar triangles. Significant computational performance gains are realized by forgoing the use of trigonometric functions in favor of similar triangles.

Using similar triangles,

$$\frac{y_s}{SSD - z_s} = \frac{y_d}{SDD} \tag{2.23}$$

$$y_d = SDD\frac{y_s}{SSD - z_s} \tag{2.24}$$

The x-coordinate of the ray-detector intersection, x_d, is calculated in the same manner. The process may be visualized by considering the top-down view of the inspection geometry, and constructing the projection triangle similar to as-shown for the y-coordinate using the side-view.

$$x_d = SDD\frac{x_s}{SSD - z_s} \tag{2.25}$$

This process is repeated for all voxels, and all exposures. The resulting quadruple-loop (three spatial directions plus exposures) is easily parallelized within a shared-memory parallelization environment such as OpenMP or CUDA/OpenCL. A distributed-memory parallelization environment, such as MPI, requires significant care to be taken in order to minimize inter-process communication. This approach is known as “voxel-centric” and was chosen to facilitate implementation in massively-parallel computing environments, such as CUDA/OpenCL. By designing the algorithm to be well-suited for execution on a graphical processing unit (GPU), the cone-beam reconstruction algorithm described here can perform the calculations several hundred times faster on a GPU than on a single CPU. Calculations which used to require hours can now be completed in less than one minute. A comparison of reconstruction accelerations achieved at CNDE through the use of GPU computing is in table 2.1.

GPU performance is very sensitive to the details of the algorithm implementation and configuration parameters passed to the device. In order to support the variety of GPU devices in active use at CNDE, routines were developed to automatically determine suitable values for the runtime parameters based on the characteristics of the device used and the calculation to be performed. These routines operate without user intervention and are discussed in appendix B.

Table 2.1 CT reconstruction times.

Hardware	**Time**	**Cost**
Serial CPU	8 hours	\$200 one-time
Quad-Core CPU, 4 threads	2 hours	\$200 one-time
64-node Beowulf cluster	12 minutes	\$15k one-time + \$5k annually
Mid-Level GeForce GPU	1.6 minutes	\$200 one-time
1st-generation Telsa GPU (C1060)	1.5 minutes	\$1500 one-time
2nd-generation Telsa GPU (C2075)	1 minute	\$2000 one-time
3rd-generation Telsa GPU (K20c)	30 seconds	\$3000 one-time

An alternative approach, known as "ray-centric" first defines a ray which connects the source to a particular detector pixel, and then proceeds to determine which voxels are intersected by this ray and perform the summation accordingly. Although each ray may be calculated independently, it is difficult to efficiently ensure that calculations associated with separate rays do not attempt to modify the contents of a shared voxel at the same time.

The quality of the reconstruction depends directly on the ability to quantify the position of a voxel within the sample, (x_s, y_s, z_s), relative to the intrinsic coordinate system defined by the source and detector. This is because the measured values on the detector which get back-projected using equations 2.25 and 2.24 are themselves a function of the material which lies along the ray-path. Thus, the back-projection must be performed using the same ray-paths.

As an example of this, consider the effect of using a fan-beam reconstruction algorithm with data acquired in a cone-beam inspection. In such a scenario, the vertical component of the radiation cone, angle β in figure 2.9, is ignored and the beam is considered to be two-dimensional. This geometry is correct in the center of the radiation cone, in what is known as the "principal fan beam". The principal fan beam is outlined in red in figure 2.9 and spans angle α. Regions within the sample which reside close to the principal fan beam can often be satisfactorily reconstructed using the fan-beam assumption. Portions of the sample which are far from the principal fan beam, however, are subject to reconstruction artifacts resulting from the assumed geometry deviating significantly from the actual geometry.

An example of this can be seen in figure 2.18. The sample being imaged is a coffee stir-stick containing spherical glass beads with a nominal diameter of 0.2 millimeters. Notice the clarity of the reconstruction in the central region of the sample and the distortion which occurs in the outer extremities.

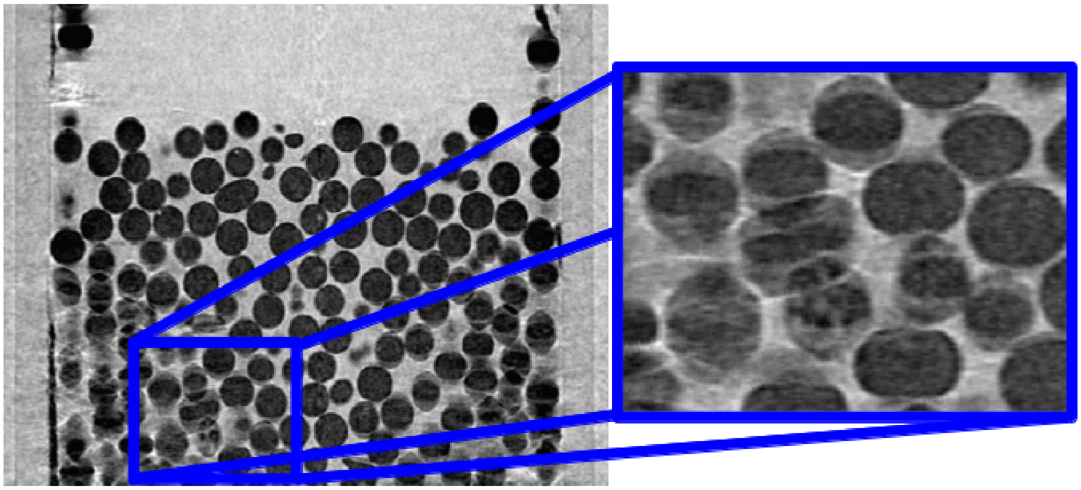


Figure 2.18 Example of artifacts produced by a fan-beam reconstruction algorithm when the data acquisition was done with a cone-beam geometry. A correct reconstruction of the same sample is shown in figure 2.19. Sample is a coffee stir-stick containing 0.2 millimeter diameter spherical glass beads.

Schematically, the source of this artifact can be seen in figures 2.20 and 2.21. In each of these figures a ray is shown traversing the sample through a particular point. In figure 2.21, the position of the ray drawn in figure 2.20 is shown in its corresponding orientation. Due to the use of a cone-beam inspection geometry the two rays do not traverse the same path and this path-divergence is the source of the artifact seen in figure 2.18.

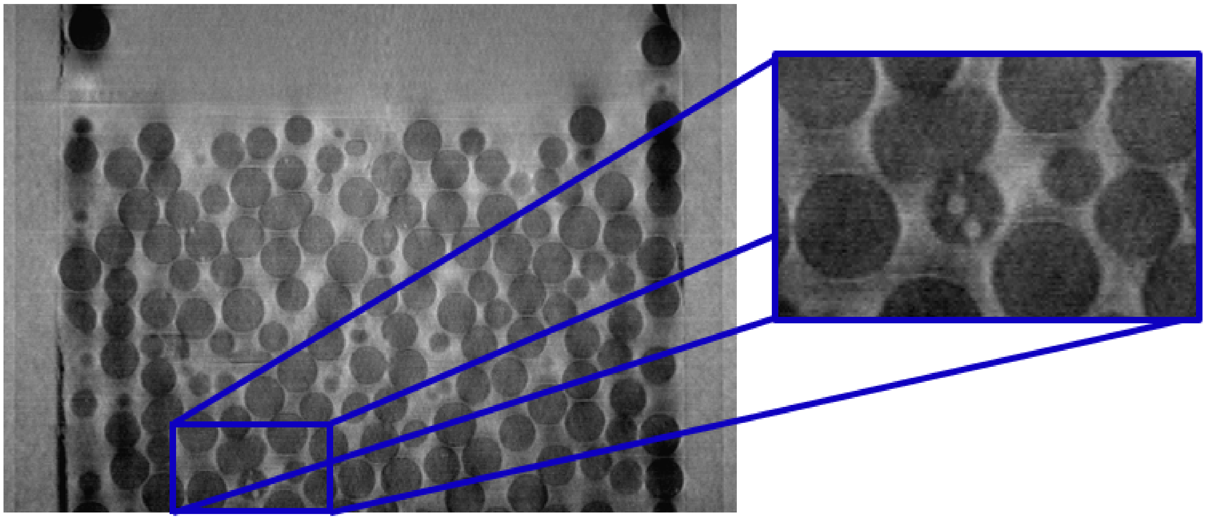


Figure 2.19 Correct reconstruction of sample shown in figure 2.18. Sample is a coffee stir-stick containing 0.2 millimeter diameter spherical glass beads.

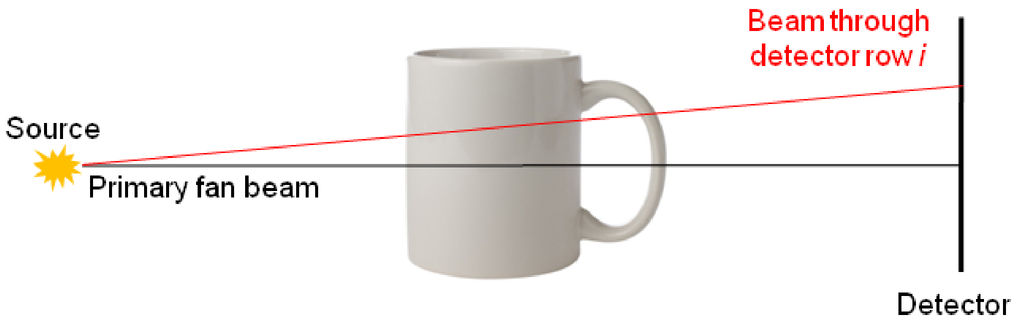


Figure 2.20 Example cone-beam geometry exposure at 0 degrees. The red line represents the interrogation volume for a particular row of pixels on the detector.

2.5.2 Implementation

To fully-illustrate the process, consider the reconstruction of a two-dimensional plane. When looking at the inspection geometry from the top-down, the setup will appear as shown in figure 2.22. Note that the global axes are shown to indicate the correct axis directions, although the global origin is still centered on the detector despite the axes being shown off-center in the interest of simplifying the sketch. Also, this example is for a simple geometry although the concepts are implemented for translation and rotation along all three axes.

The process begins by discretization of the reconstruction volume into voxels. For this illustration the voxel colored red will be the current voxel of-interest (figure 2.22). Creating

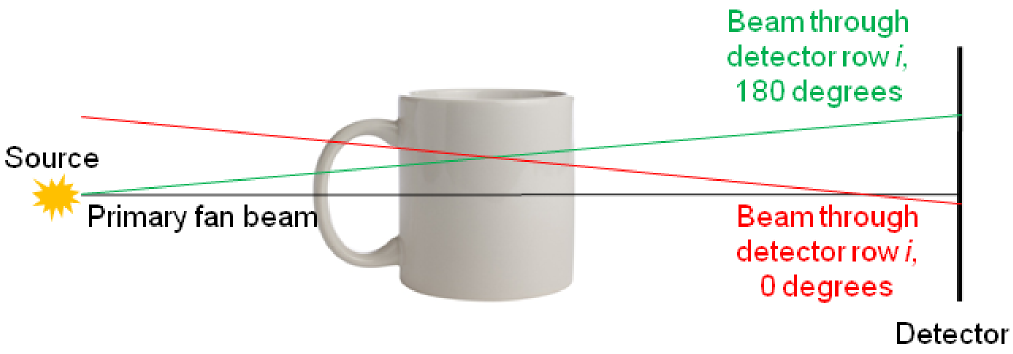


Figure 2.21 Example cone-beam geometry exposure at 180 degrees. The red line represents the same interrogation volume as shown in figure 2.20 while the green line represents the interrogation volume for the same detector row, but with the sample rotated 180 degrees. Notice how the lines do not lie on top of one another, illustrating how the interrogation volume changes as the sample is rotated during the inspection.

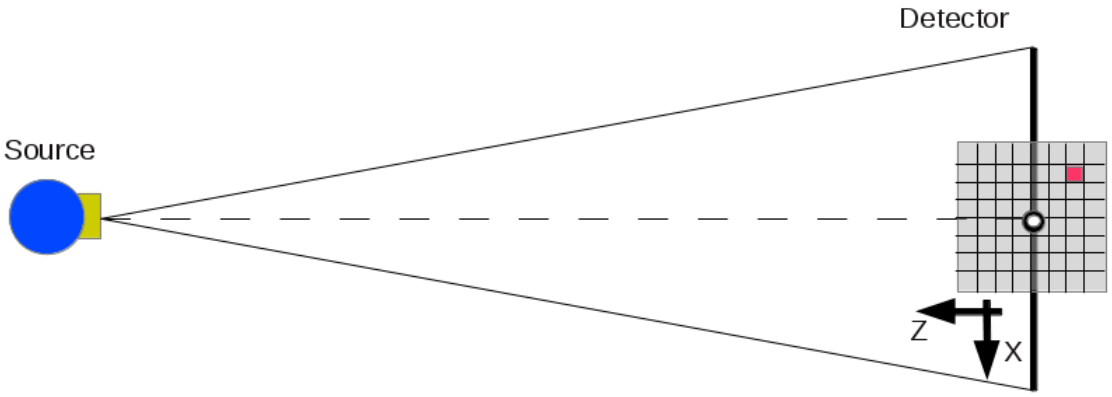


Figure 2.22 Back-projection calculations, step 1. The reconstruction grid is generated with the sample's coordinate axes coinciding with the global coordinate axes.

the entire reconstruction volume entails performing this process for each voxel independently.

With the discretized volume still centered on the global origin, the voxel's coordinates are calculated after applying the specified rotations. In this case there is a simple rotation about an axis coming out of the page. Use of equation 2.26 allows these rotations to be performed about all three axes, and account for any misalignment between the rotation axes and the global axes (figure 2.23).

As shown in figures 2.23 and 2.24, the sample rotation is calculated prior to translation. This is done simplify the rotation calculations. The rotation is defined about the origin of the

```
/* Calculate (x, y, z) coordinate of voxel at indices (idx_x, idx_y, idx_z). */
x = xmin + dx*idx_x;
y = ymin + dy*idx_y;
z = zmin + dz*idx_z;
```

Listing 2.1 Discretize the reconstruction volume.

sample-centric coordinate system, requiring that any relative displacement is considered when applying the specified rotations. Additionally, keeping the voxel coordinate values within the global reference frame makes the code easier to understand and simplifies the back-projection calculations by avoiding the confusion of using multiple coordinate systems simultaneously.

Supporting fully-generalized rotation about all three axes, which may or may-not be aligned with the global axes uses Rodrigues' Formula (Koks (2006)) to calculate the post-rotation coordinates, **x'**, of the voxel as a function of the voxel's original position, **x**, and a rotation, θ, about an axis defined by the unit vector, **k**.

$$\mathbf{x'} = \mathbf{x}cos\theta + (\mathbf{k} \times \mathbf{x})sin\theta + \mathbf{k}(\mathbf{k} \cdot \mathbf{x})(1 - cos\theta) \tag{2.26}$$

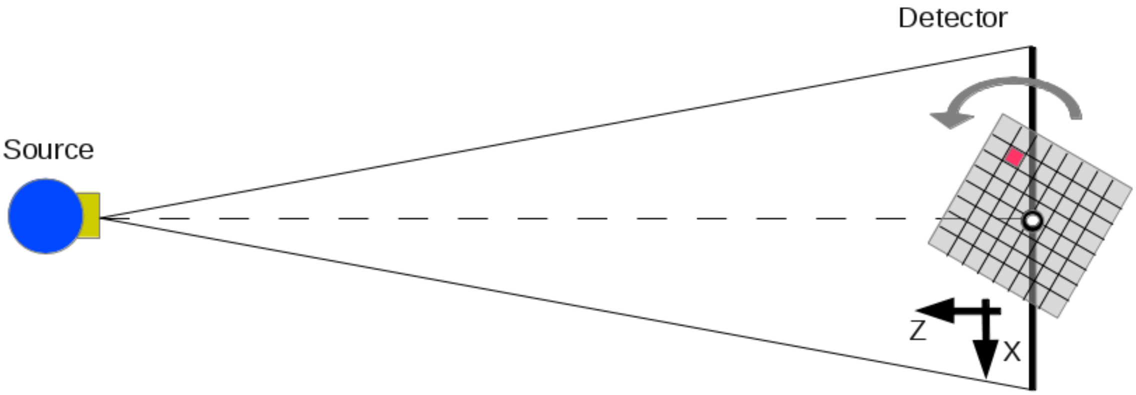


Figure 2.23 Back-projection calculations, step 2. The reconstruction grid is rotated. No translation has occurred yet.

After the rotations are applied, the rotated grid is next translated (figure 2.24). In general this includes a translation towards the source as well as a side-to-side (vertical in these top-down views) translation. The motion towards the source provides the inspection magnification, and the lateral motion corrects for the problem of having the rotation axis not precisely centered on the detector.

```
/* Perform rotations using Rodrigues' Formula.
 *
 * The rotation axes are defined by the unit vectors kx, ky, kz.
 *
 * Rotations are about the global axes in the order: X -> Y -> Z.
 */

/* Rotation about x-axis. */
d = kx[0]*x + kx[1]*y + kx[2]*z;
xp = x*cos(th_x) + (z*kx[1] - y*kx[2])*sin(th_x) + kx[0]*d*(1 - cos(th_x));
yp = y*cos(th_x) + (x*kx[2] - z*kx[0])*sin(th_x) + kx[1]*d*(1 - cos(th_x));
zp = z*cos(th_x) + (y*kx[0] - x*kx[1])*sin(th_x) + kx[2]*d*(1 - cos(th_x));
x = xp;    y = yp;    z = zp;

/* Rotation about y-axis. */
d = ky[0]*x + ky[1]*y + ky[2]*z;
xp = x*cos(th_y) + (z*ky[1] - y*ky[2])*sin(th_y) + ky[0]*d*(1 - cos(th_y));
yp = y*cos(th_y) + (x*ky[2] - z*ky[0])*sin(th_y) + ky[1]*d*(1 - cos(th_y));
zp = z*cos(th_y) + (y*ky[0] - x*ky[1])*sin(th_y) + ky[2]*d*(1 - cos(th_y));
x = xp;    y = yp;    z = zp;

/* Rotation about z-axis. */
d = kz[0]*x + kz[1]*y + kz[2]*z;
xp = x*cos(th_z) + (z*kz[1] - y*kz[2])*sin(th_z) + kz[0]*d*(1 - cos(th_z));
yp = y*cos(th_z) + (x*kz[2] - z*kz[0])*sin(th_z) + kz[1]*d*(1 - cos(th_z));
zp = z*cos(th_z) + (y*kz[0] - x*kz[1])*sin(th_z) + kz[2]*d*(1 - cos(th_z));
x = xp;    y = yp;    z = zp;
```

Listing 2.2 Rotate the reconstruction volume.

```
/* Apply sample translation */
x += transx[i];
y += transy[i];
z += transz[i];
```

Listing 2.3 Translate the reconstruction volume.

In addition to translating the volume, any detector translations and rotations are applied in this step.

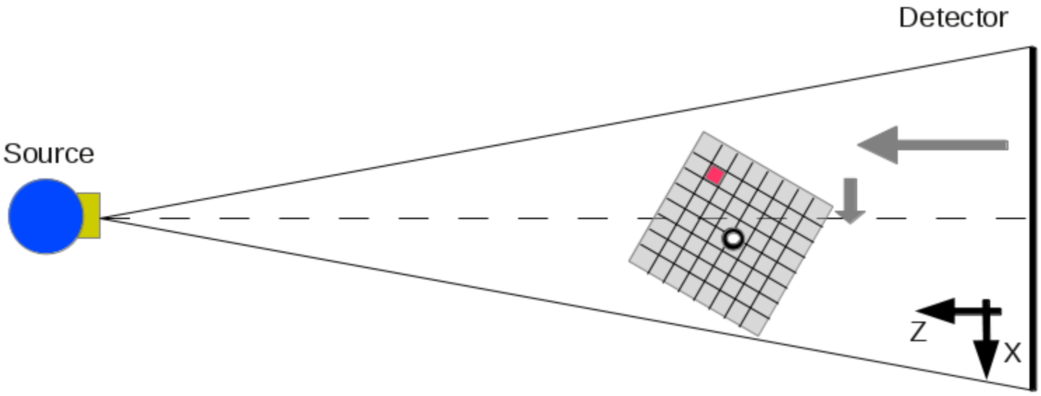


Figure 2.24 Back-projection calculations, step 3. The rotated reconstruction grid is translated.

Finally, after the rotations and translation have been applied, the actual back-projection calculations may be performed (figure 2.25) using similar triangles as described above.

This rotation-translation-projection sequence is calculated independently for each voxel and for each exposure.

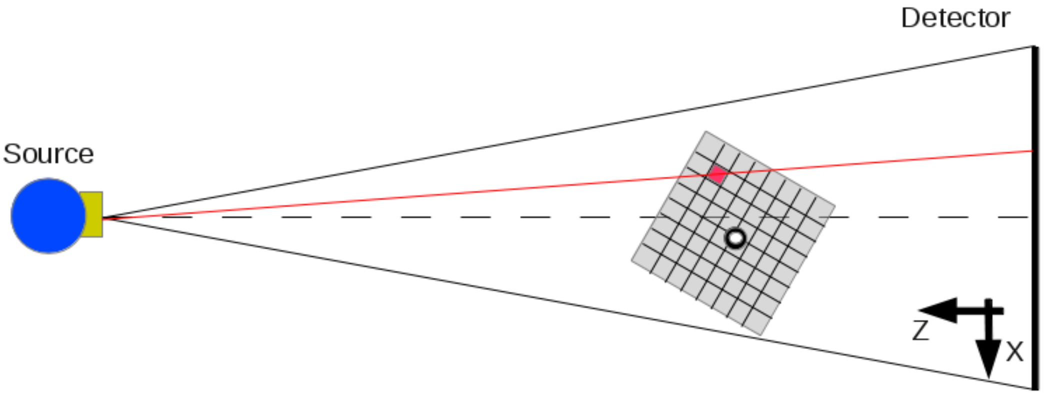


Figure 2.25 Back-projection calculations, step 4. The back-projection calculations are performed.

CHAPTER 3. RECONSTRUCTION POST-PROCESSING

The CT reconstruction process is simply the first step in the analysis tool-chain. The reconstruction volume which is created must then be further analyzed in order to extract the desired damage characterization information. This analysis is complicated by several factors, including large-magnitude reconstruction artifacts such as beam-hardening, the small-magnitude signatures produced by early-state damage, and the massive quantity of data generated by the reconstruction process. A computational analysis tool is necessary as it is impractical for human operators to manually sift through 10+ gigabytes of data per CT reconstruction.

Unfortunately, the analysis tools currently available to the X-ray imaging community rely on simple algorithms which consider each voxel separately from its neighbors. This assumption of voxel-independence is a result of the historical lack of computational power rather than the physics of the inspection. A consequence of this assumption is that small, low-contrast signals are overlooked, requiring an operator to laboriously review massive datasets. The subtle signals produced by early-stage damage require the development of sophisticated, physics-based image processing algorithms which are capable of detecting very low contrast features.

3.1 Damage Detection

The first step in characterizing damage within a material is identifying the damage regions within the CT reconstruction itself. This is where the challenge of using commonly-available polychromatic tube sources becomes apparent due to the beam-hardening artifact.

3.1.1 Polychromatic Complications

The earlier derivation in section 2.1.1 assumes that the incident radiation beam is monochromatic, which means that the x-ray photons all possess the same energy and wavelength. To use an optical analogy, the photons are all the same “color”. In a typical laboratory setting using a tube source, the incident beam is polychromatic, meaning the photons have a spectrum of energies, or colors, like a white light. A proper derivation would be required to consider the energy-dependence of the sample's attenuation, as-shown in equation 3.1

$$I = \int_0^{E_{max}} I_0(E) e^{-\int \mu(E,l)dl} dE \tag{3.1}$$

where E_{max} is the maximum photon energy produced by the tube.

Algebraic and statistical reconstruction algorithms can be designed to accommodate this polychromatic behavior. When the more-simple back-projection methods are used, failure to address the polychromatic nature of the beam produces an artifact known as “beam hardening”. Beam hardening is caused by the low energy photons being absorbed in the outer regions of the sample, thereby hardening the beam as it passes through the sample. When the x-ray detector is simply counting photons, and not discriminating by photon energy, this leads to the reconstruction result appearing as if the outer edges are higher-attenuating than the central region. This affect can be seen in figure 3.1. The bowl-shaped trend in the line-trace, shown by the green line, is the beam hardening artifact. The bowl-shaped trend introduced by this artifact must be removed prior to applying statistical analysis routines for feature extraction.

3.1.2 Beam-Hardening Correction

Recall equation 2.2

$$I = I_0 e^{-\int \mu(l)dl}$$

which is the foundation of back-projection reconstruction algorithms. As noted in section 3.1.1, the attenuation coefficient, μ, is a function of photon energy. Thus, the more-complete version of equation 2.2 is found in equation 3.1.

$$I = \int_0^{E_{max}} I_0(E) e^{-\int \mu(E,l)dl} dE$$

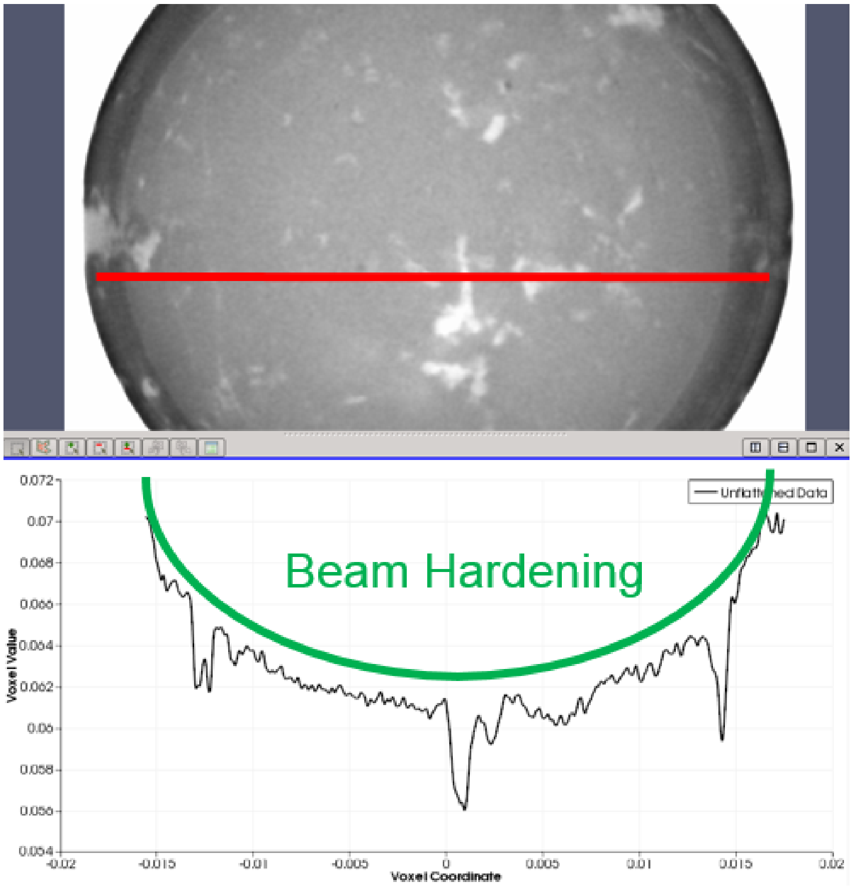


Figure 3.1 Beam-hardening reconstruction artifact. The bowl-shaped trend in the line-trace, shown by the green line, is the beam hardening artifact.

Common flat-panel detectors, such as those used at CNDE as well as countless other facilities, function by simply counting x-ray photons. More photons, caused by a larger value of I in equations 2.2 and 3.1, produce a larger signal which is digitized by the detector and then interpreted by the data acquisition workstation. These detectors do not measure the energy of the incoming photons. Although energy-sensitive detectors exist, they are designed for point-measurements rather than 2D arrays and their use in CT would require a laborious acquisition process reminicent of the first CT scanner developed by Hounsfield in the early 1970's.

A consequence of the energy-dependence of the attenuation coefficient, μ, is that the low energy photons are preferentially absorbed in the outer regions of the sample. As the beam passes through the sample these low-energy photons are absorbed in the near-surface region near the source while the higher-energy photons continue to pass through the sample. When the reconstruction is performed, the result reflects this larger attenuation of the outer regions and introduces a long-length trend in the reconstruction result.

The removal of low-energy photons from a beam is known as "beam hardening" and is this is the source of the artifact name. When performing qualitative analysis with the human eye (e.g., "just looking" at the result) it is possible for a trained operator to simply disregard the beam hardening artifact. However, when attempting a quantitative analysis this artifact introduces significant complications.

The simplest method of addressing this artifact is to flatten the image by removing the long-length trends from the result. This flat-fielding operation results in a stationary mean background value and is required for the statistical analysis algorithms. An example of flattening the dataset can be seen in figure 3.2. A bilateral filter with a large kernel was used to determine the long-length trends.

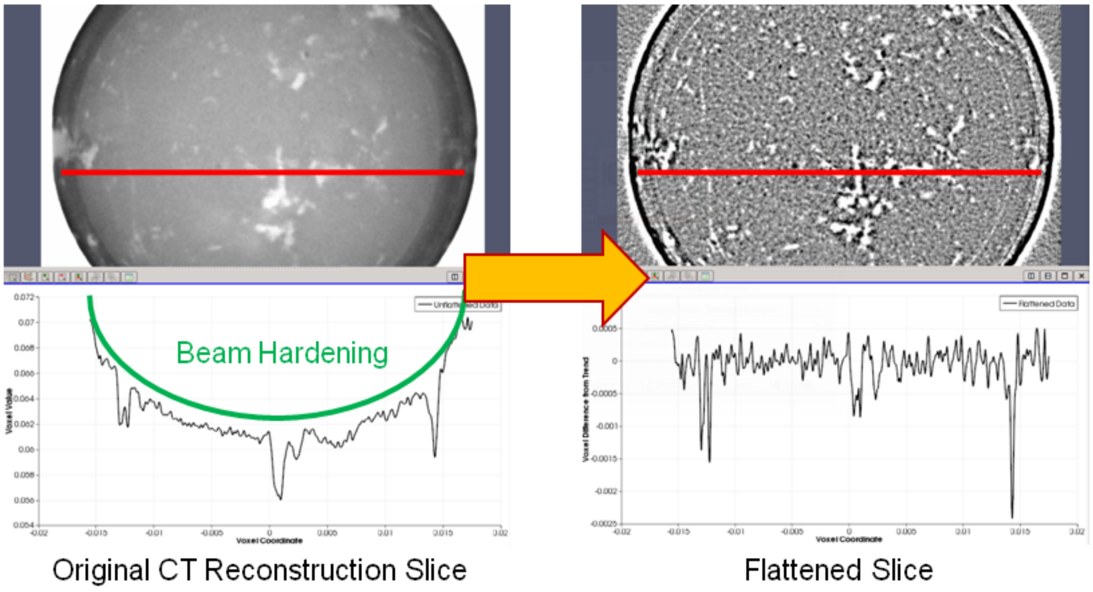


Figure 3.2 Illustration of trend-removal operation used to remove the beam hardening reconstruction artifact.

3.1.3 Damage Detection

In the simplest cases, regions of the reconstruction volume corresponding to damage can be identified by setting a single threshold. This approach requires the reconstruction to be high-contrast in order to have a reliable discrimination between the material itself and the damage structures.

In many cases, however, the signals produced by the damage are subtle, and the contrast between those signals and the base material is insufficient for adequately defining a threshold. This is especially the case when attempting to study the early indications of damage where the signals are very small due to the small size of the damage.

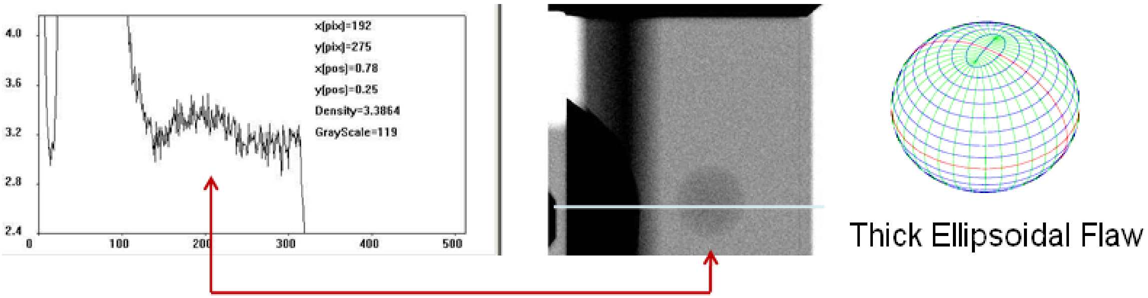


Figure 3.3 Simple, low-contrast flaw readily visible to the human eye but difficult to find using a basic threshold. Notice that the defect will be under-sized by if threshold is set to minimize chance of false-calls, while an accurate size measurement can only be captured by a threshold which will produce many spurious indications due to the noise magnitude being of the same order as the ellipsoid's contrast. Ellipsoid from http://geographiclib.sourceforge.net

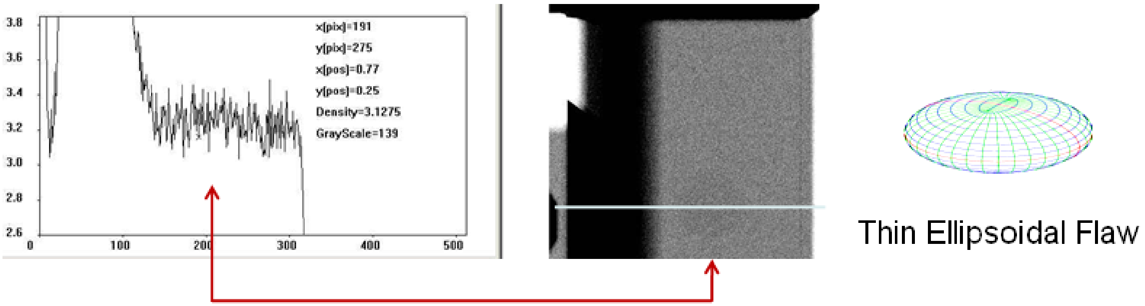


Figure 3.4 Simple, low-contrast flaw visible to the human eye but impossible to find using a basic threshold. Notice that the magnitude of the noise fully-obscures the ellipsoid's contrast. Ellipsoid from http://geographiclib.sourceforge.net

The limitation of using a simple threshold arises from the fact that it considers each reconstruction voxel independently of all the others. When the human eye identifies a damage region, however, it is not because a single voxel met a particular criteria. Human operators readily identify damage because their eyes see a *region* of voxels, which are all different from the background material. Their minds recognize that it is very unlikely that a cluster of pixels would deviate from the background in a unified manner.

This observation indicates that the ability to reliably identify damage regions can be greatly improved by leveraging the statistical behavior of a neighborhood of voxels. By considering a voxel's value in the context of its neighbors it is possible to develop a much more robust algorithm for identifying damage while reducing the input and effort required of a human inspector (Gray et al., 2004; Grandin and Gray, 2014).

This reduction of human interaction is very important. The analytical capability of a human inspector is very impressive, but it is also variable. Even the best inspector will have good days and bad days, and the capability will necessarily differ between inspectors. The existence of such variability introduces an additional source of certainty which increases the time and cost required to properly quantify the damage initiation and evolution of a material.

Human inspectors also improse another constraint: they are slow. As x-ray detector technology and computational power increases, so does the size of the datasets to be processed. With the advent of GPU computing the ability to generate CT reconstructions far outpaces capacity for analyzing them. Current CT datasets at CNDE are on the order of 10 gigabytes, and that will continue to rise. This problem is not unique to CT, either. Phased-array ultrasound can easily produce datasets just as large and other non-destructive techniques are rapidly increasing their own data quantities.

When it is desired to have fast, consistent completion of a task it is natural to turn to a computational solution. The primary challenge introduced by using a computer to perform the analysis is developing an algorithm which can approach the analytical ability of a human inspector. As illustrated earlier, a simple threshold does not meet that criteria. However, development of a statistical analysis can bring the analysis-quality criteria into the grasp of a computer algorithm.

3.1.3.1 Binomial Hypothesis Analysis

Statistics based algorithms function by considering ensembles of data values rather individual data points. A robust defect detection algorithm based on hypothesis testing assuming a binomial distribution was demonstrated by Gray et al. (2004). The algorithm begins by considering two distributions of data values: a reference region and a test region. The reference region is typically large and defines the background noise present in the data. This region is boxed in red in figure 3.5. The test region is typically much smaller and is swept through the data. This region is boxed in blue in figure 3.5.

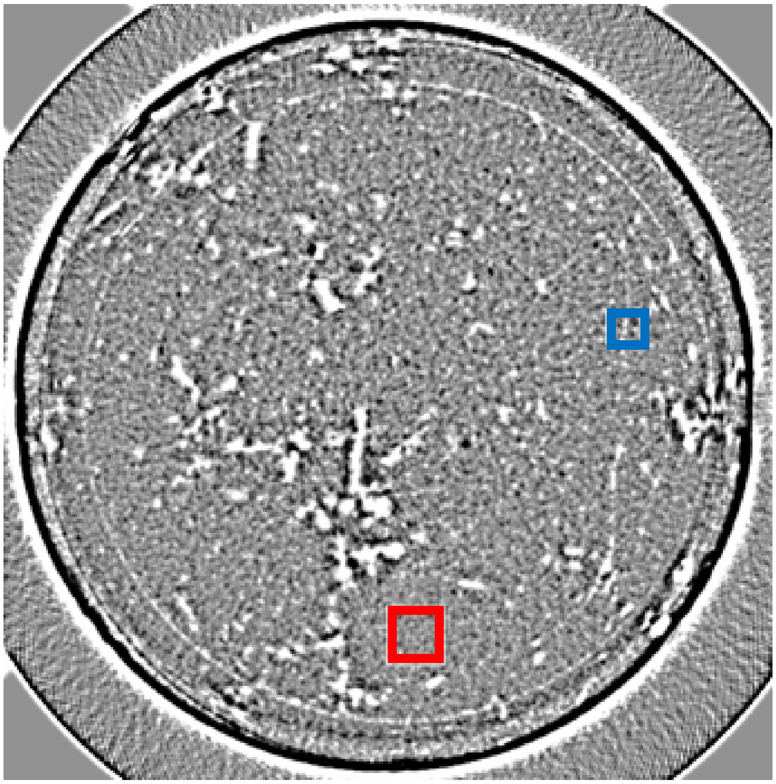


Figure 3.5 Flattened reconstruction slice of dolomite rock core showing the reference and test regions used by the Binomial Hypothesis analysis. The test region is contained by the blue box and the reference region is contained by the red box.

During the analysis the reference region remains in a fixed position while the test region is swept through the dataset. For each location of the test region, the histograms of the reference and test region are compared. A sample set of histograms can be seen in figure 3.6.

The histograms in figure 3.6 are clearly different. The reference region, in red, is more-tightly clustered around an expected value while the test region, in blue, has a tail to the left (i.e., is "left-skewed"). Applying a hypothesis test based on a binomial distribution allows one to calculate the likelihood that the test region contains an observation of the background noise

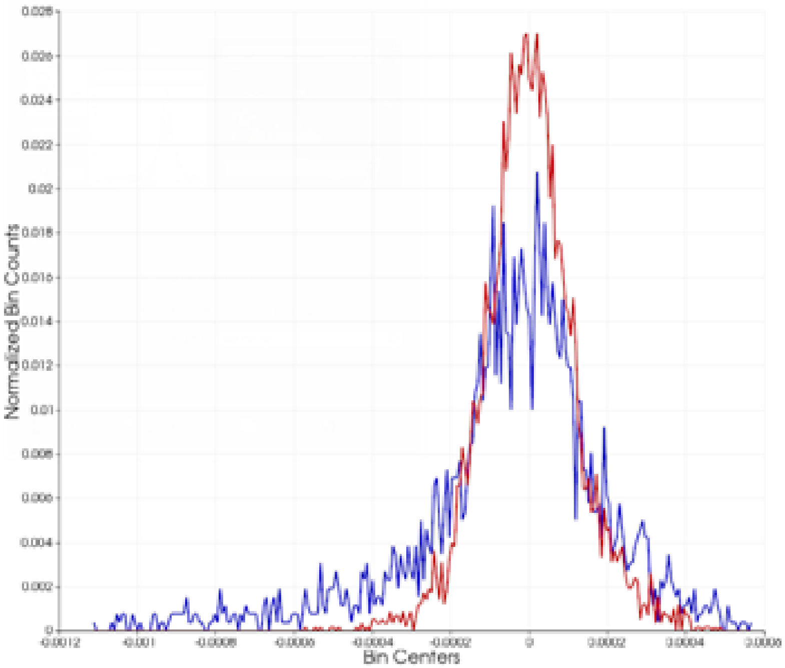


Figure 3.6 Histograms produced by each of the boxed regions in figure 3.5. Again, blue corresponds to the test region and red to the reference region.

rather than an observation of an internal feature. In figure 3.6 the left-skew indicates that the test region contains an internal feature, and in this case it's a pore. The pore is small, but can be seen in figure 3.5 as the small white portion within the blue box.

3.1.3.2 Kolmogorov-Smirnov Analysis

An alternative analysis can be performed using what is known as the Kolmogorov-Smirnov (KS) statistical test. The KS test quantifies the similarity of two distributions by finding the maximum difference between their cumulative probability distributions (CDFs). When sampled data is used, rather than an analytic description of the distribution, the empirical distribution function (EDF) is used.

Since probability functions are being used, the results are constrained to have a value between 0 and 1, with 0 indicating that the two distributions are identical and 1 indicating that the distributions are very different. This allows for the ability to set reliable thresholds at which a region is identified as containing damage. The openended-ness of the binomial hypothesis analysis makes the reliable determination of such a threshold very difficult.

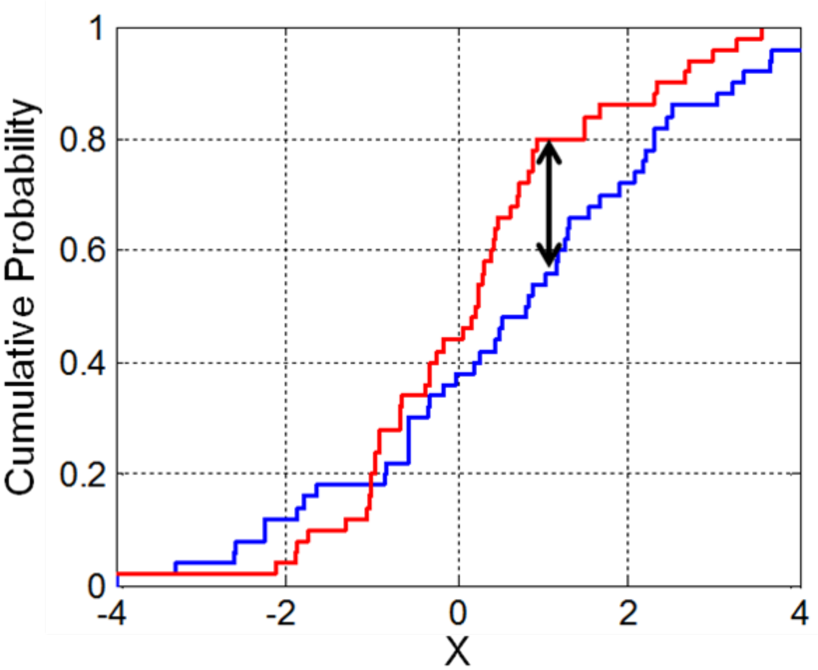


Figure 3.7 Illustration of Kolmogorov-Smirnov (KS) statistical test. The red and blue lines are the empirical distribution functions (EDFs) for two different samples, and the black arrow indicates the maximum difference between the two EDF curves. Typical useage of the KS test only considers the magnitude of the difference, but additional information may be obtained by considering the sign of the difference. Note that the distributions shown in this figure do not correspond to the red and blue boxed regions in the previous figures.
Image taken from http://en.wikipedia.org/wiki/File:KS2_Example.png

Typically, the KS test only considers the magnitude of the maximum difference between the two distributions. It can be extended, however, to retain the sign of the maximum distance. This causes the output to be bounded by [-1, 1] rather than [0, 1]. As before, near-zero values indicate a great similarity between the two distributions. However, the signed distance indicates the difference in the mean between the reference region and the test region. This permits further analysis steps to focus on a particular type of damage. For example, a single sweep of the signed-KS analysis will flag regions of low-density with the reconstruction, such as porosity and cracks, as well has regions of high-density, such as embedded contaminants and precipitates.

The typical, unsigned-KS analysis cannot distinguish between these different classes of damage and defects. The binomial hypothesis analysis can be tailored to identify each type of damage, but multiple sweeps of the data are required to cover each type.

3.1.3.3 Binomial Hypothesis vs. Kolmogorov-Smirnov

The performance of each analysis type is very similar. This is to be expected since they are fundamentally performing the same analysis, but with each algorithm framed in a different manner. A qualitative comparison of the algorithm performance is shown in figures 3.8 and 3.9. In this figure, the statistical analysis results have been thresholded to delineate between likely-defect and likely-background regions. The area which meets this threshold criteria is then translucently overlaid on the original image in order to show the capability of each algorithm.

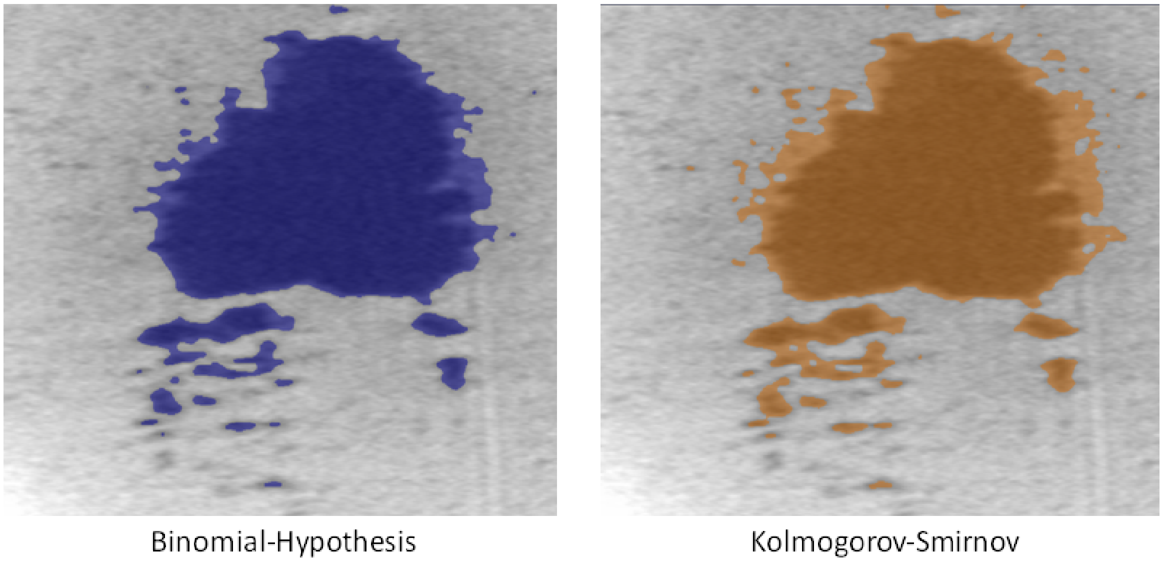


Figure 3.8 Qualitative comparison of binomial hypothesis and Kolmogorov-Smirnov algorithm performance using a powder metallurgy sample. The differences between them are explained by the selected value of the likelihood threshold which separates likely-defects from likely-background.

Although the analytical performance is equivalent, the KS analysis provides numerical implementation advantages

1. Bounded results. The results are bound to the intervals [-1, 1] (signed variation of the analysis) and [0, 1] (typical, unsigned implementation). This allows for the determination and implementation of reliable thresholds which can be used in later analysis steps.

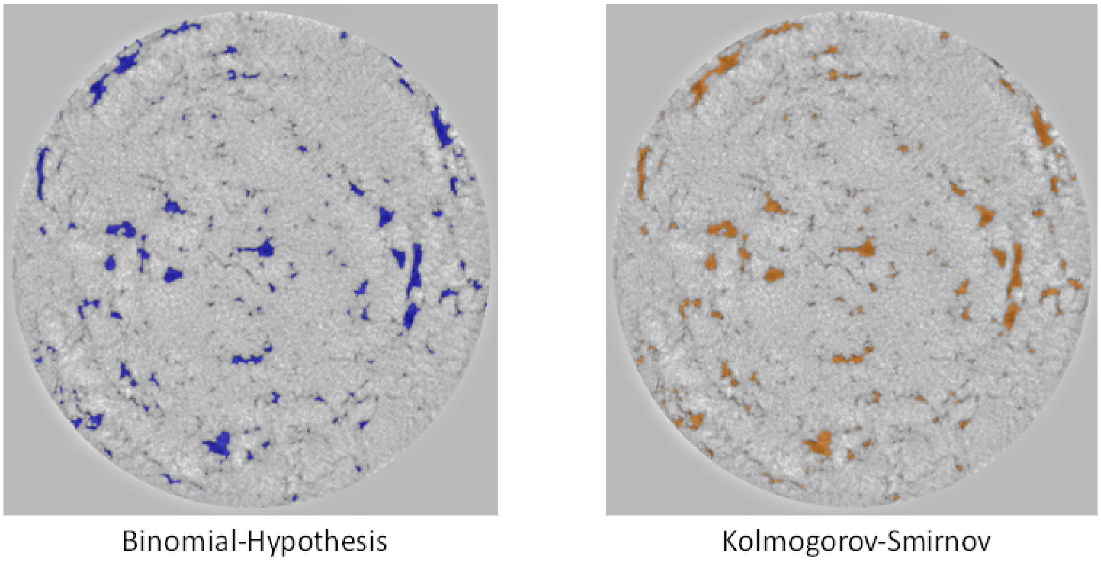


Figure 3.9 Qualitative comparison of binomial hypothesis and Kolmogorov-Smirnov algorithm performance using a dolomite rock core. The differences between them are explained by the selected value of the likelihood threshold which separates likely-defects from likely-background. The background image of the dolomite core sample has been de-trended.

2. Execution speed. Due to the implementation details, the binomial hypothesis analysis requires the recalculation of several quantities. The KS analysis implements those quantities in such a way as to only require their calculation once. This results in a significant time-savings, especially on large, 3D reconstruction volumes. The binomial hypothesis routine requires nearly 40 minutes to process an 800 x 800 x 40 volume. The KS analysis requires 30 seconds to perform the same analysis.

In both algorithms, volumetric features such as porosity are readily captured. Narrow, linear features such as cracks prove more difficult, and improved crack detectability is left for future work.

3.2 Data Segmentation

The above-described statistical analysis algorithms serve to increase the contrast of a dataset. The task of segmentation, identifying localized regions of connected voxels as a single logical entity, has yet to be performed. Segmentation routines are most-effective on high-contrast data, and the statistical analysis routines improve the contrast of relevant features in preparation for the segmentation process iteslf.

After the statistical analysis has been performed, the reconstruction volume can now be thought of as a point-cloud of likelihood values, where large magnitudes indicate likely damage and low magnitudes indicate likely to be the base material. Before meaningful quantities may be calculated, however, this point-cloud dataset must be further processed to determine the logical entities which the human brain interprets as the individual instances of damage. When a human operator evaluates the results of the statistical analysis he or she doesn't see independent voxels, but rather clusters of similar likelihoods. A cluster of high-likelihood values is interpreted as a single logical entity, an instance of damage or a defect. The next step in the analysis chain is to cause the computational algorithm to perform the same task and identify the various clusters of voxels which are all part of the same instance of damage. This task of partitioning the dataset into multiple homogeneous regions is known as data segmentation.

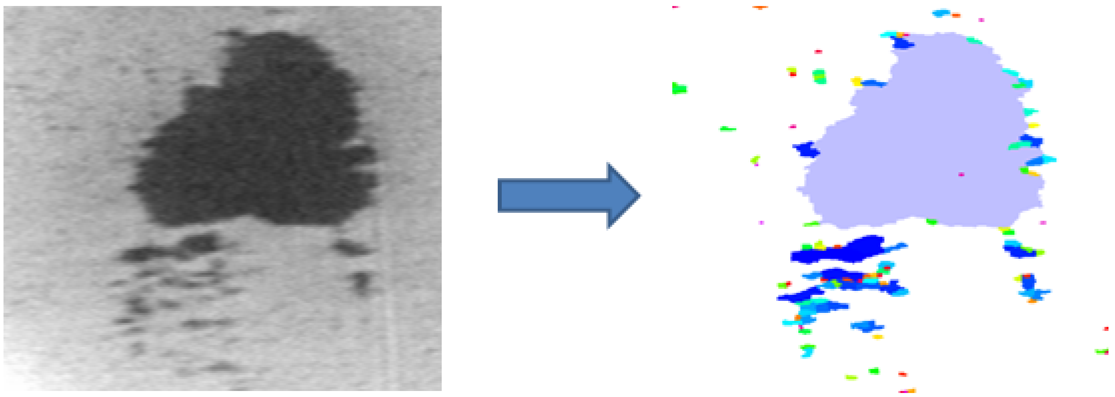


Figure 3.10 Input and output of a segmentation routine. The image on the left is a slice from a CT reconstruction while the image on the right is colored to identify the segmented regions identified within the region. Each single-color region indicates a collection of pixels which have been grouped together into the same logical entity, assorted pores in this case. This particular dataset is high-contrast due to the size of the pores, but the same process can be applied to low-contrast data after applying the statistical algorithms.

Data segmentation is an active research field of its own and has previously been probed by CNDE (Sheikh, 2006). For ease of implementation, this thesis makes use of the Insight Toolkit (ITK) which provides an open-source library of several data segmentation and registration routines tailored for processing image data (Johnson et al., 2013).

In particular, this work makes use of the watershed segmentation capability within ITK. Other options are also available within the toolkit, and the watershed algorithms were chosen due to their comparative robustness with respect to user-selected parameters.

Watershed segmentation operates on the gradient of a dataset and assumes large gradient magnitudes occur at segment boundaries while small gradients exist within segments. In low-contrast data these assumptions are often false and the statistical analysis algorithms described above are used to increase the contrast and improve the segmentation performance. This behavior is illustrated in the high-contrast figure 3.11. Calculating the gradient, like any derivative operation, is sensitive to noise, so a smoothing operation is often employed to lessen the effects of image noise. The noise suppression must be balanced against the blurring of edges between segments.

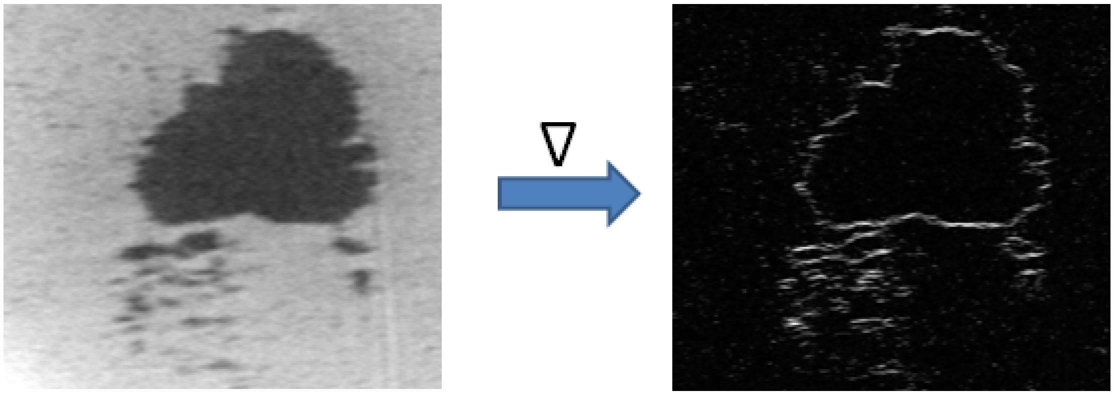


Figure 3.11 Preparing input data for watershed segmentation. The image on the left is a slice from a CT recosntruction and the image on the right is the magnitude of the 2D gradient of that image. Large gradients occur at the boundaries between segments.

The watershed algorithm gets its name from the terrain-contour like behavior of the image gradient. If one envisions plotting the gradient in figure 3.11 as a 3D surface with the gradient magnitude defining the height of the surface, it can be seen that the surface would contain a series of catchment basins. If water were to be poured over the surface a discrete collection of puddles would form, and the quantity of puddles would be determined by the depth to which the basins are filled. Small quantities of water would fill numerous puddles, many of them small and separated by low ridges. As more water is added, the smallest of the puddles overflow their boundaries and join with their neighbors to form a reduced number of larger puddles. Naturally this analogy may be carried to its extreme end when enough water has been added to overflow *all* ridges, but there will be a water level for which the small puddles caused by image noise have been joined into larger puddles, and these larger puddles identify the structural elements of the image.

Within the image segmentation nomenclature, the water level controls whether the image becomes over-segmented or under-segmented. Over-segmentation occurs when too many discrete segments have been identified. This corresponds to the water analogy in the case where only a small amount of water has been added and there are several small, shallow puddles created. In reality these puddles, the segments, are members of the same logical entity in the image, but image noise has caused there to be small ridges in the gradient which leads to the single entity being broken into multiple pieces.

Under-segmentation occurs when there are too few segments. This corresponds to the case of over-filling the basins in the water analogy. In this case the ridge created by a true boundary has been overflowed and the distinction between the two enties in the image has been lost in the segmentation.

The ITK toolkit refers to this fill-depth parameter as the "level", and its effect can be seen in figure 3.12. Figure 3.12 contains a line-trace through the gradient image of figure 3.11. The trace was taken at the bottom of the large pore in image 3.11. In the top image of figure 3.12, the level-parameter was set to 50% of the maximum gradient height while in the bottom image the parameter was set to 25%. In this example it is easy to see how the lower parameter will produce an over-segmented result while the high parameter produces a better result. In this line trace the pore is split into two segments due to the line-trace catching the small peak in the non-flat bottom of the pore.

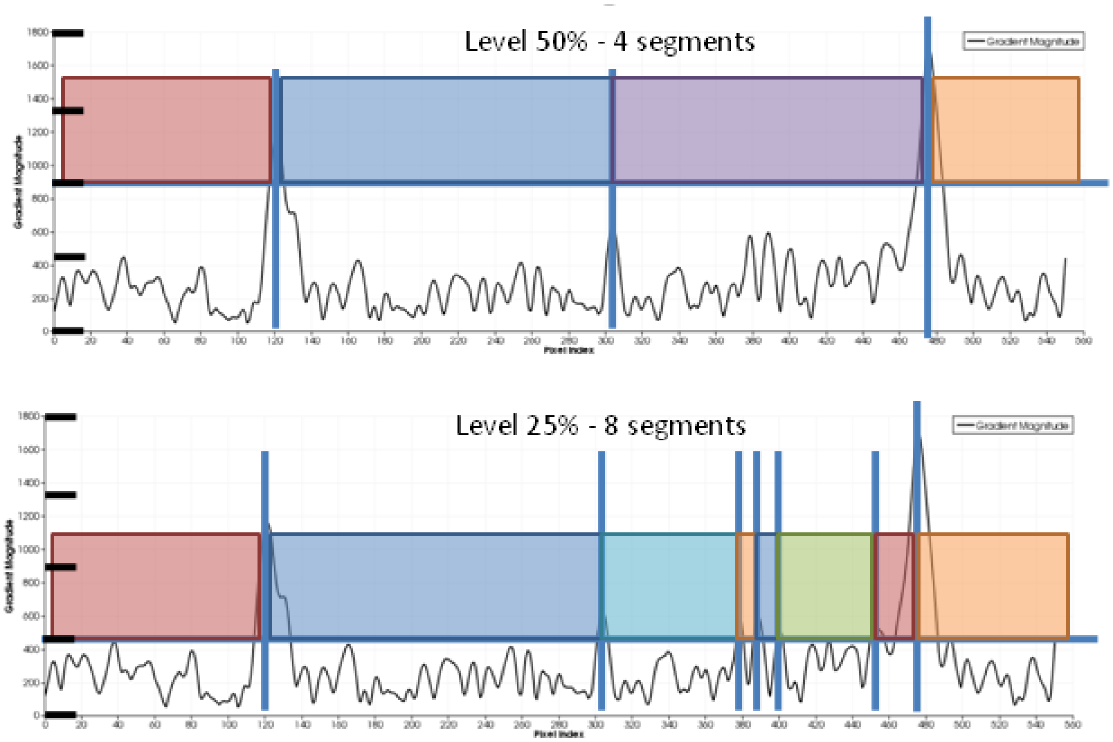


Figure 3.12 Illustration of the effect of the "level" parameter for watershed segmentation. Lower level values tend towards producing an over-segmented image where a single feature is erroneously broken up into multiple pieces, while larger level values tend towards under-segmenting the image where multiple features are combined into a single segment.

CHAPTER 4. APPLICATION TO FIBER REINFORCED COMPOSITES

As described in section 1.3, multiple research groups have come very close to imaging fiber-scale features within fiber reinforced composites. These studies stopped short of studying damage structures between individual fibers and focused instead on the larger ply and tow structures. Studying features on that length-scale, however, does not allow one to study the early stages of damage which are crucially important for developing structural health and prognosis models. Studies investigating matrix cracking due to impact damage, radio-opaque contrast agents were injected into the material to enhance the image quality. Reliance on contrast agents requires that the damage structures be interconnected so that the contrast material may be brought into the entirety of the damage structure. Further, the injection of contrast agents can affect the mechanical behavior of a material, rendering their use unsuitable for studying the evolution of damage structures.

4.1 Fiber-Scale Resolution

Using the exponential-decay window function described in section 2.4.2 the noise properties of the reconstruction have been improved and allow low-contrast signals to be seen. As a result, it is now possible to image fiber-scale structures within fiber reinforced composites. Examples of this can be seen in figures 4.1 - 4.6.

Achieving crisp images of individual fibers remains a challenge, which is expected considering the experimental setup. These scans were performed in the high-resolution CT system at CNDE using a magnification factor of 45, which is the maximum-allowable for the current system configuration. This resulted in the flat-panel detector, a GE DXR 500L which has a pixel pitch of 100 microns, having an effective pixel pitch of 2.22 microns, which allows two pixels per fiber. Ideally, three or more pixels would be required since having a feature span at least three pixels allows the signal to begin at a background level, change due to the feature, and return to the background level. This limit is exceeded in CT, however, due to the large numbers of projection images acquired during the scan. As a result, it is possible to see indications of features which are too small to be crisply imaged. For these particular scans, the sample was rotated 0.25 degrees between each exposure, resulting in a total of 1440 projection images being acquired for each scan.

An additional complicating factor when attempting to image individual fibers is the finite size of the source. The control software for the x-ray source reported the focal spot to have a diameter of 14 microns. With an effective detector pixel pitch, and a matching reconstruction voxel size, of 2.22 microns, the scan is pushing far beyond the typically-accepted geometric unsharpness limits. The geometric unsharpness for these CT scans would be calculated to be over 600 microns, or approximately 6 pixels on the detector. At 45x magnification, these 6 pixels span 13 microns.

The contrast between fibers and matrix is large-enough in these figures to allow a simple threshold to segment the two materials within the composite. This high contrast is a direct result of the improved PSF window function introduced in section 2.4.2, and is significant due to the current challenge of identifying narrow, linear features when using the statistical post-processing algorithms, as stated in section 3.1.3.3.

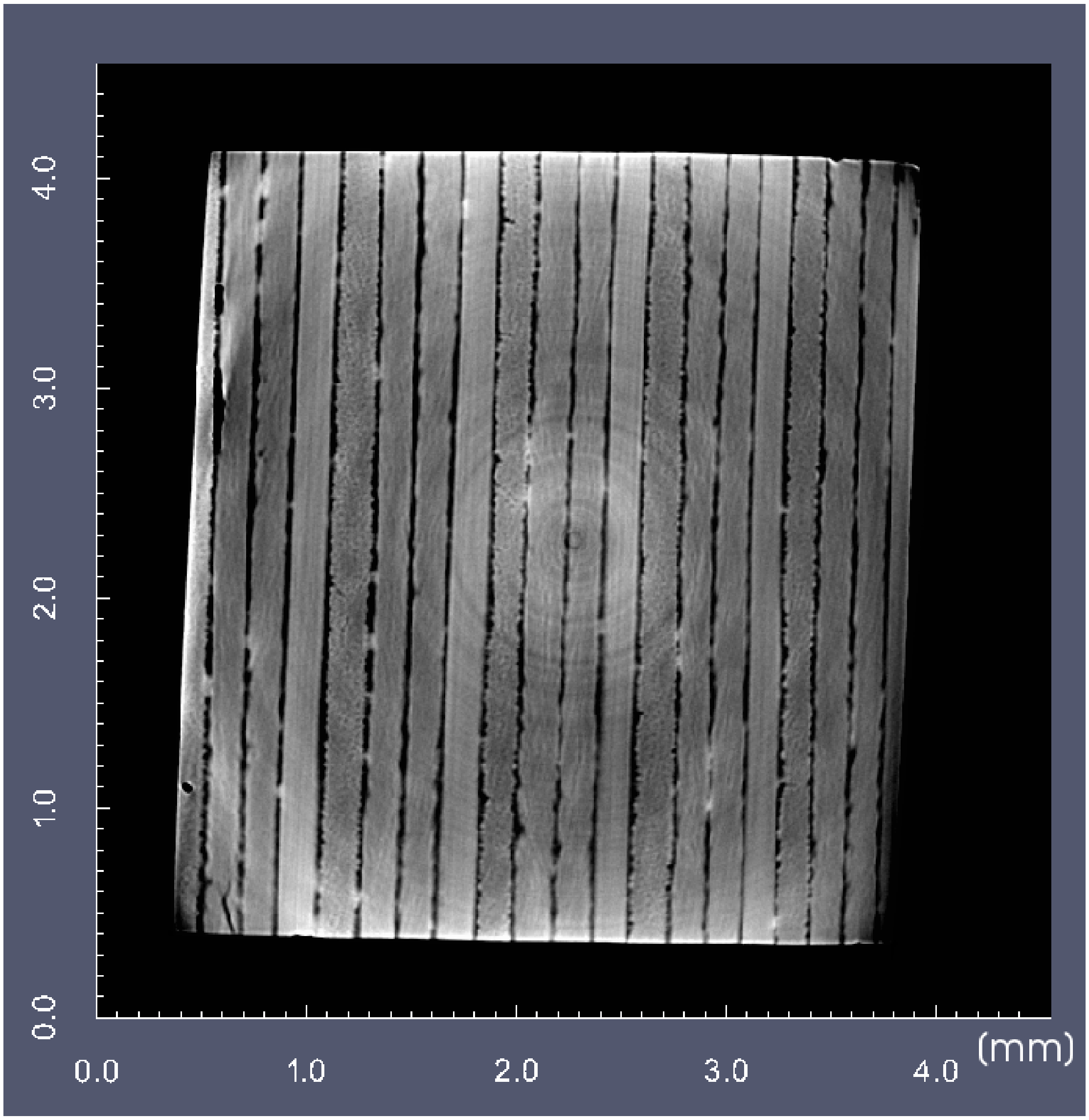


Figure 4.1 Example CT reconstruction slice demonstrating granule-scale resolution in a carbon fiber reinforced polymer composite. The sample has a quasi-isotropic $[0/+45/-45/90]_s$ layup, with the 0-degree fibers coming out of the page clearly visible. The light-colored plies to the left of the 0-degree plies contain fibers oriented at 90-degrees and the ±45-degree plies are to the right of the 0-degree plies. Notice the fiber-scale detail which is visible, particularly in the 0-degree plies. The faint, concentric rings are an artifact caused by the x-ray detector used in the CT scan.

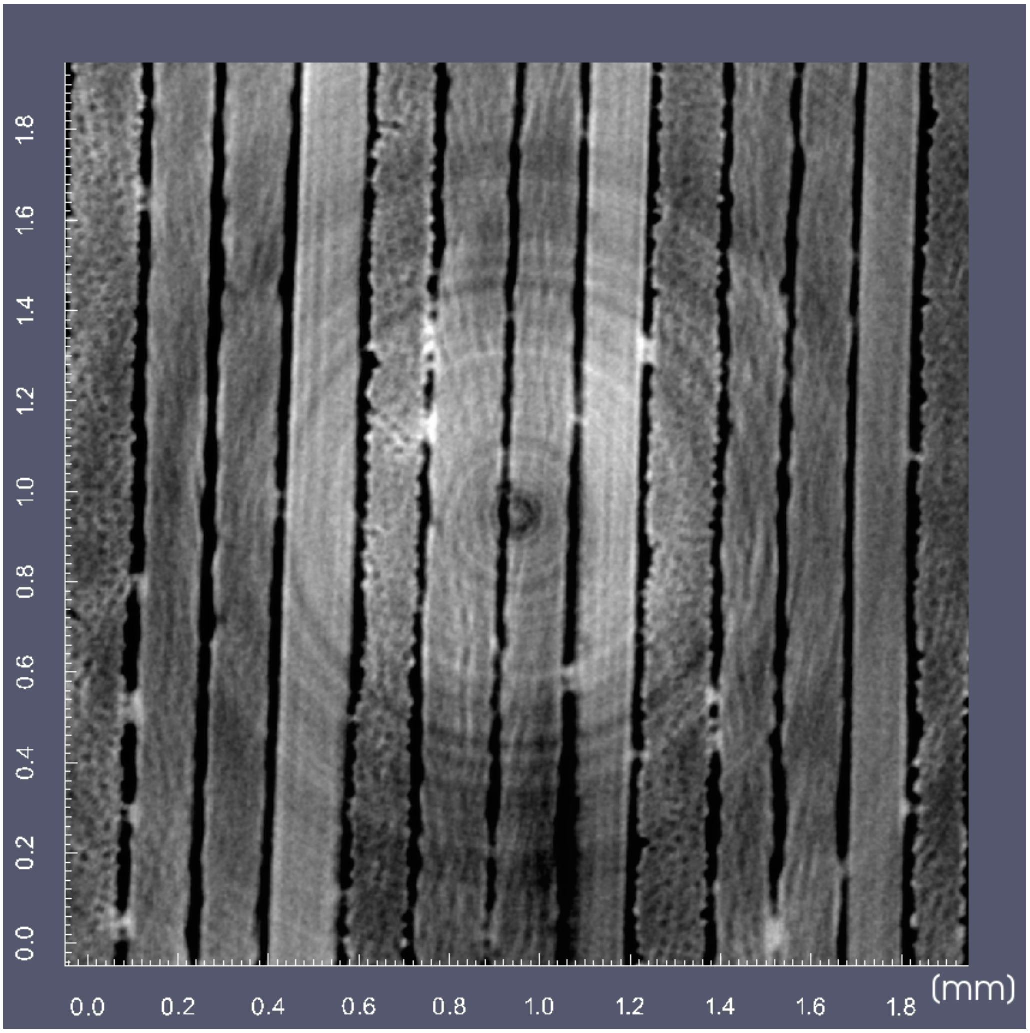


Figure 4.2 Central region of figure 4.1. Voxel size has been reduced to 2 microns. Fiber diameter is 4 microns.

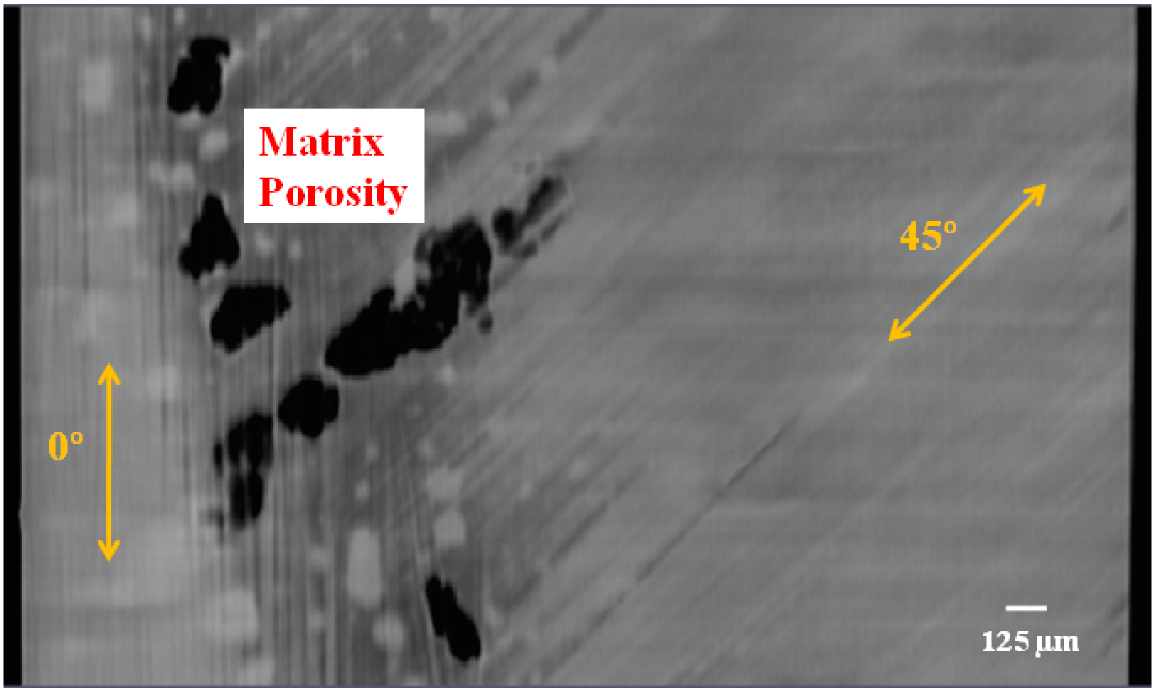


Figure 4.3 Porosity within matrix between plies. Multiple fiber orientations are seen because the reconstruction grid is not aligned with the plies.

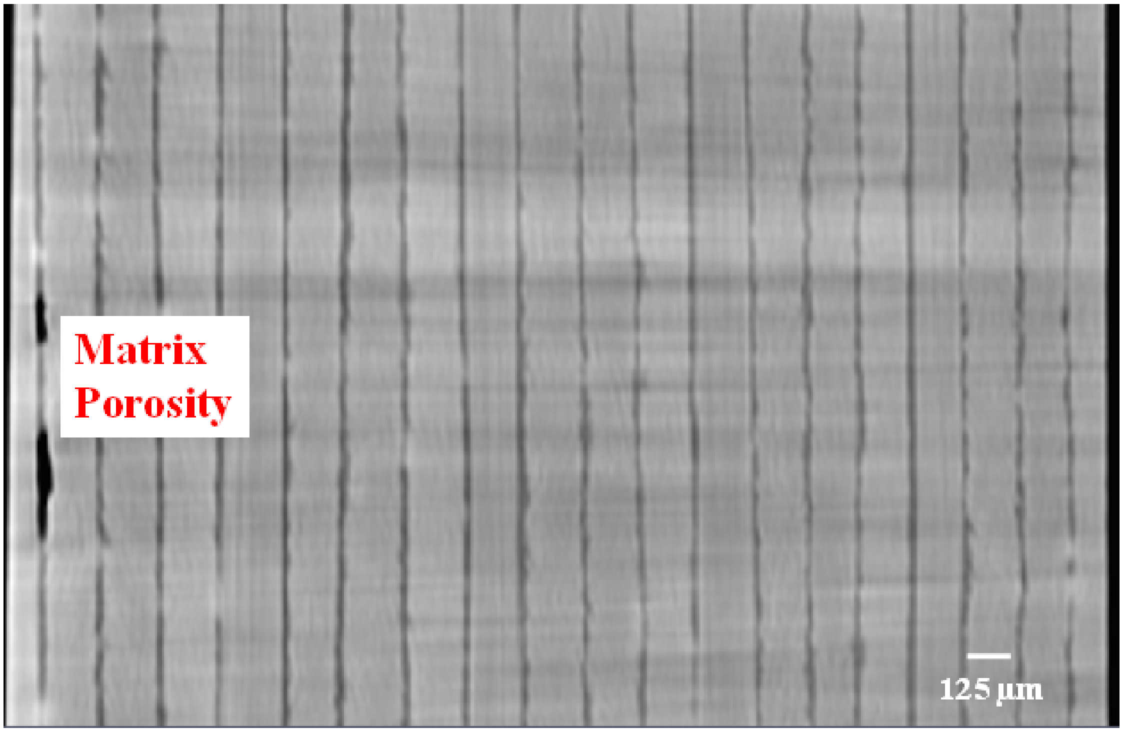


Figure 4.4 Side-view of matrix porosity between plies. Porosity shown in figure 4.3 is on the left side of the image.

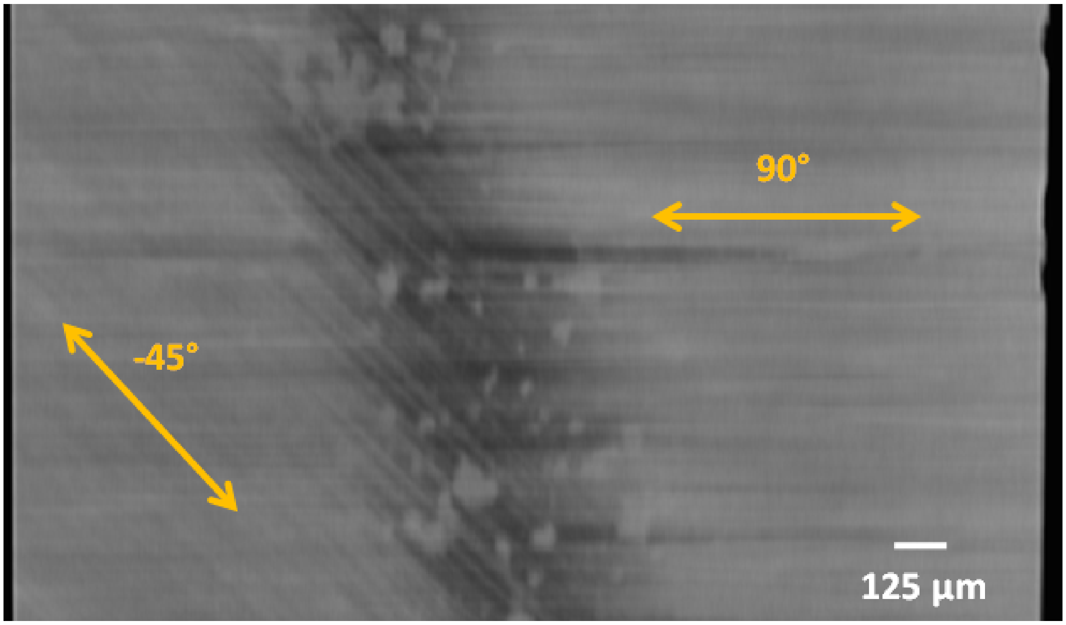


Figure 4.5 High-density inclusions within matrix between plies. Multiple fiber orientations are seen because the reconstruction grid is not aligned with the plies.

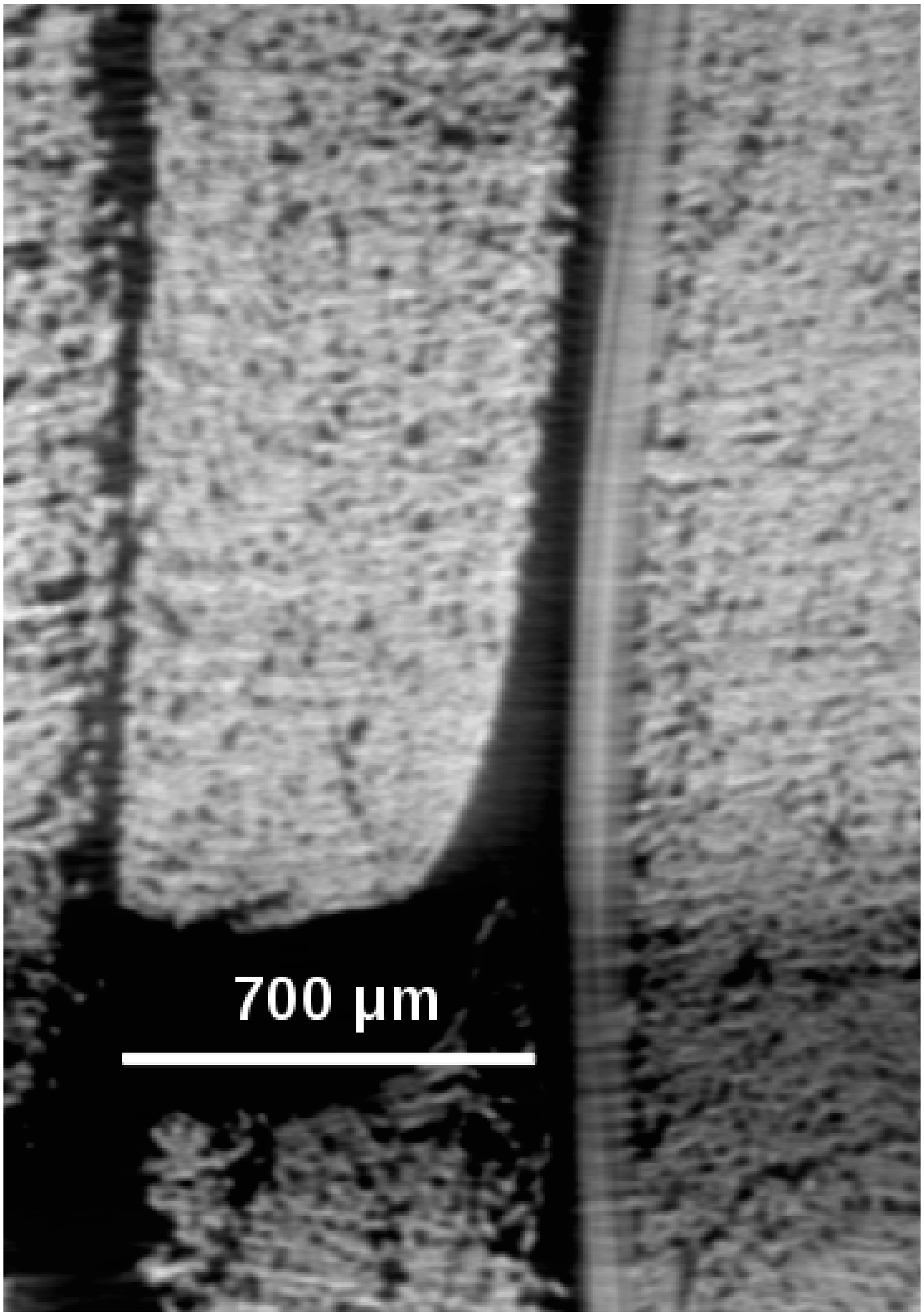


Figure 4.6 CT reconstruction slice from glass fiber reinforced polymer composite. Voxel size is 2 microns. Fiber diameter is 4 microns.

4.2 Damage Evolution

An important motivation for improving the imaging capability of the high-resolution CT system at CNDE is the need to study how damage is initiated and how it evolves over time. Development of useful structural health and prognosis models requires a detailed understanding of the material's granule-scale behavior as it is subjected to loads.

To demonstrate the ability to identify and track damage over time, a series of CT scans was performed on a narrow strip of carbon fiber reinforced composite, shown in figure 4.7.

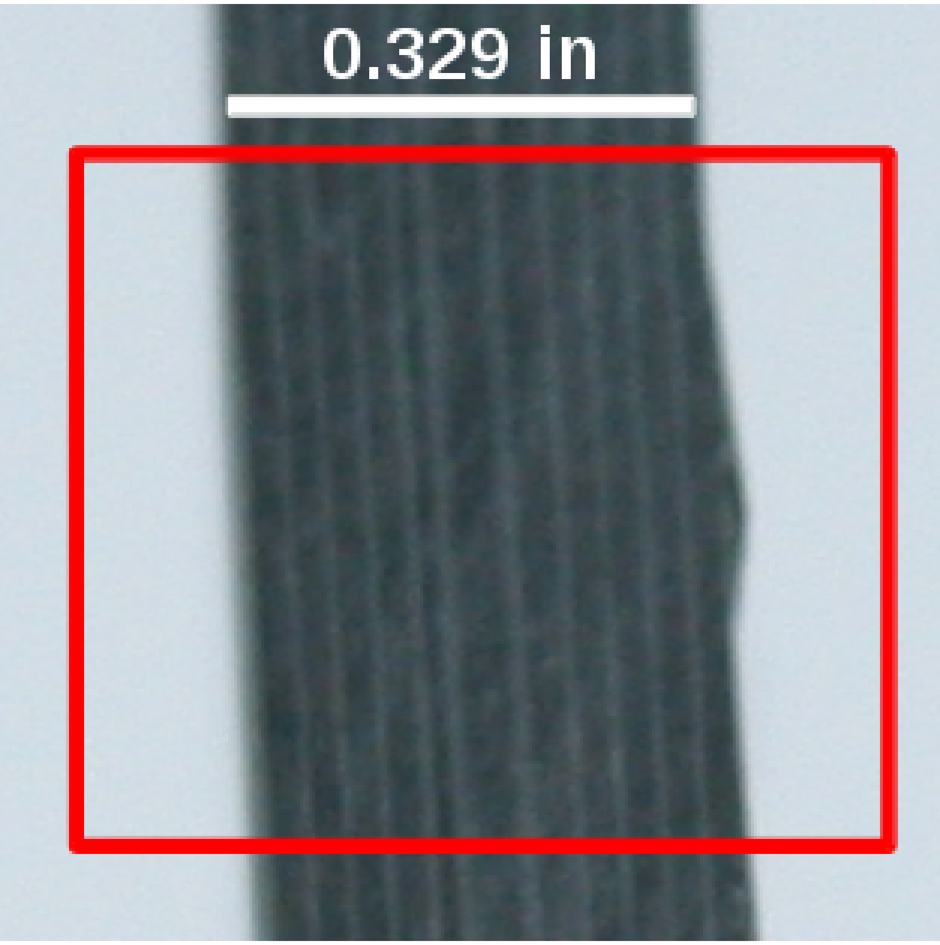


Figure 4.7 Photo of damage-evolution sample. The sample itself is 6 inches long, 0.25 inches wide, and 0.3 inches thick. It was cut from the center of a larger, 6 x 2 x 0.3 inch coupon.

The sample had a quasi-isotropic layup, $[0/\pm45/90]_s$, consisting of 48 plies. Each ply was separated by a layer of toughening rubber. This toughening layer is shown schematically in figure 4.8 and with optical microscopy in figure 4.9.

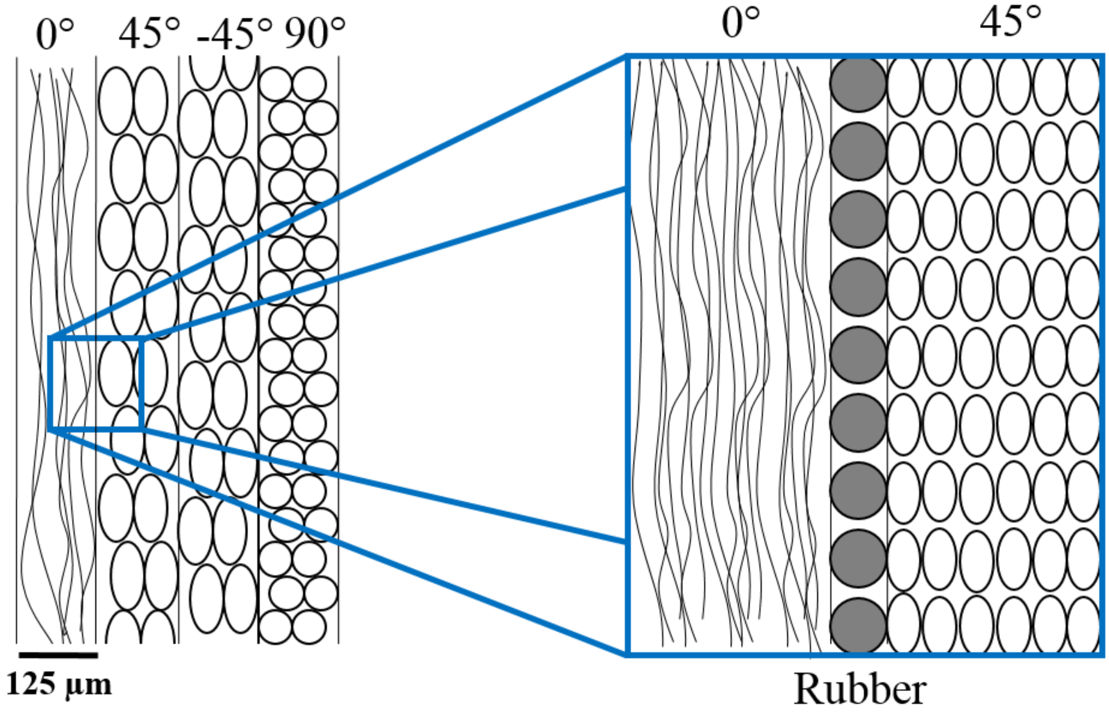


Figure 4.8 Sketch of carbon fiber reinforced polymer test specimen. Note the presence of the toughening rubber between plies.

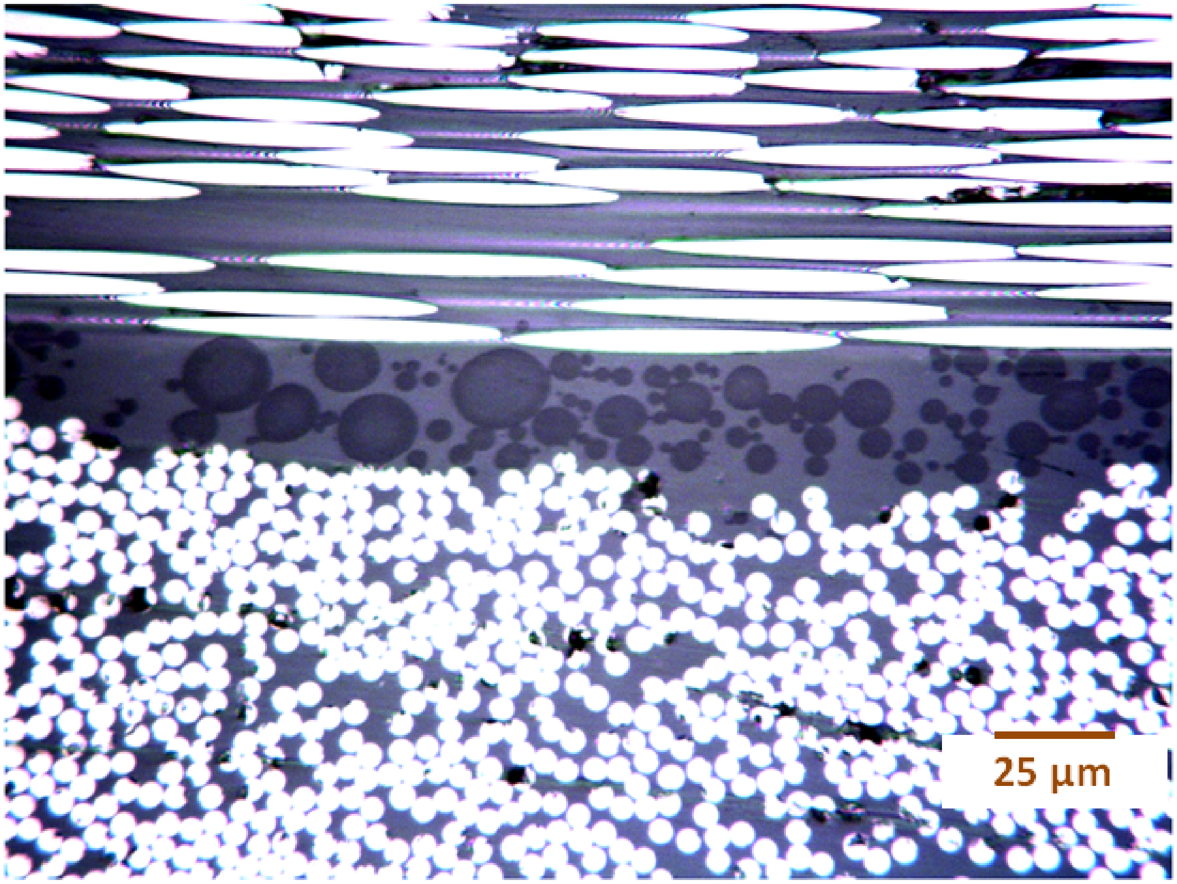


Figure 4.9 Micrograph of ply interface in carbon fiber reinforced polymer test specimen. Note the random distribution and sizes of the rubber particles. The image contrast has been stretched to make the rubber particles easier to see. The interface shown is between fibers oriented at 90-degrees (top) and 0-degrees (bottom).

4.2.1 Initial Damage

Initial damage consisted of buckling of the outermost ply due to the load applied in a 4-point bend test. This buckling occurred at the contact point between the sample and the roller used as part of the test apparatus, identified as region “3” in figure 4.10.

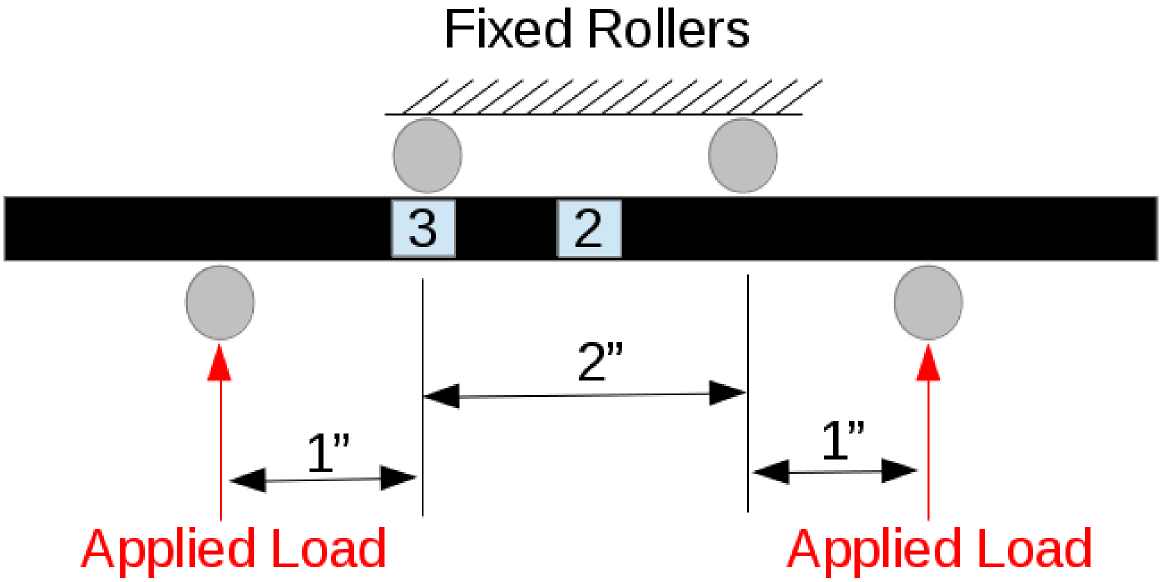


Figure 4.10 Sketch of 4-point bend setup. The region of interest is identified by the number “3”, and corresponds to the region just below one of the fixed upper rollers.

This initial damage was compounded by subjecting the sample to additional loads in the MTS machine at CNDE. The MTS machine was programmed to perform a uniaxial push with a displacement rate of 1x10^{-4} inches per second. The intent was to induce an incremental increase in the quantity of internal damage within the sample and a very slow displacement rate was necessary to ensure that the force could be removed before catastrophic damage could occur.

An acoustic emission (AE) probe was used to help guide the damage application. Once the load within the sample exceeds the strength of the material, fracture occurs as the sample attempts to shed the load (Subramanian, 2013). As the fibers and matrix fracture they emit ultrasonic waves which are measured by a probe which simply “listens” to the sample. A significant shortcoming of the AE measurement is that it provides no localization of the source of the signal, so even though AE signals may be recorded which indicate damage has occurred it cannot be known if the damage has occurred within the region of interest.

The sample is nearly silent as the load is first applied since the initial loads are well within the strength of the material. When a load-shedding event occurs there is a burst of ultrasonic activity and then a return to near silence. When inducing damage on the sample the monotonic MTS program was executed until two such bursts of ultrasonic emissions were observed. Initial tests showed that removing the applied load after a single burst failed to induce an appreciable amount of damage.

Figure 4.11 shows the load applied by the MTS machine and the energy of the ultrasonic emissions, both as functions of time. The cyclic behavior of the MTS force is due to its being programmed to control displacement rather than force. When the sample softens, such as after a load-shedding event, the displacement will over-shoot the programmed position and the MTS machine will pull-back to get back on the programmed profile. Additionally, the MTS machine was "tuned" for applying cyclic fatigue loads, and this is a significant source of the periodic structure in the force data.

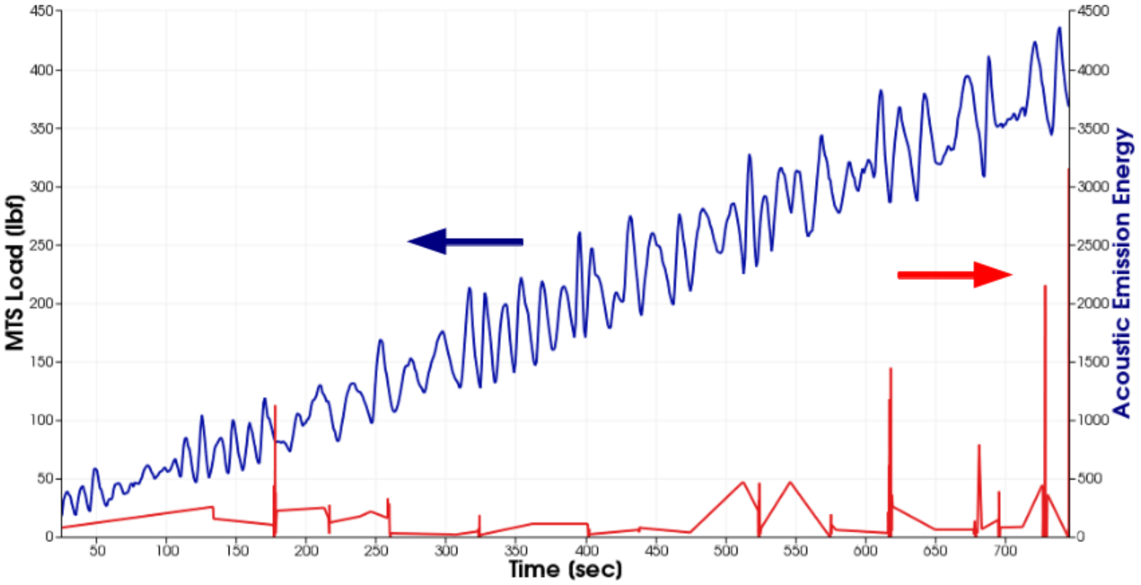


Figure 4.11 MTS and Acoustic Emission data from first round of damage-induction. Applied load by the MTS is in blue and is read off the left axis. Acoustic energy measured by the Acoustic Emission probe is in red and read off the right axis.

A CT scan was performed on the sample immediately after its removal from the MTS machine. Unfortunately, the sample did not remain steady during this scan, resulting in the inability to achieve the desired image quality for studying the damage structures, as shown in figure 4.12. it is believed that the unsteadyness is the result of the sample relaxing residual internal loads during the duration of the 18-hour scan. It is noteworthy that the bulk of the sample can be crisply resolved, including the irregular ply-boundary in the center of the sample. This clarity suggests that only a portion of the sample, specifically the portion containing the damage structures, was unsteady during the scan, and this supports the hypothesis that the sample underwent relaxation during the scan. The displacement which occurred is very small, as evidenced by the double-imaging having a relative displacement on the order of a single ply. Due to the large magnification of the scan (approximately 14x), a seemingly-small, unquantified motion such as this can introduce significant artifacts.

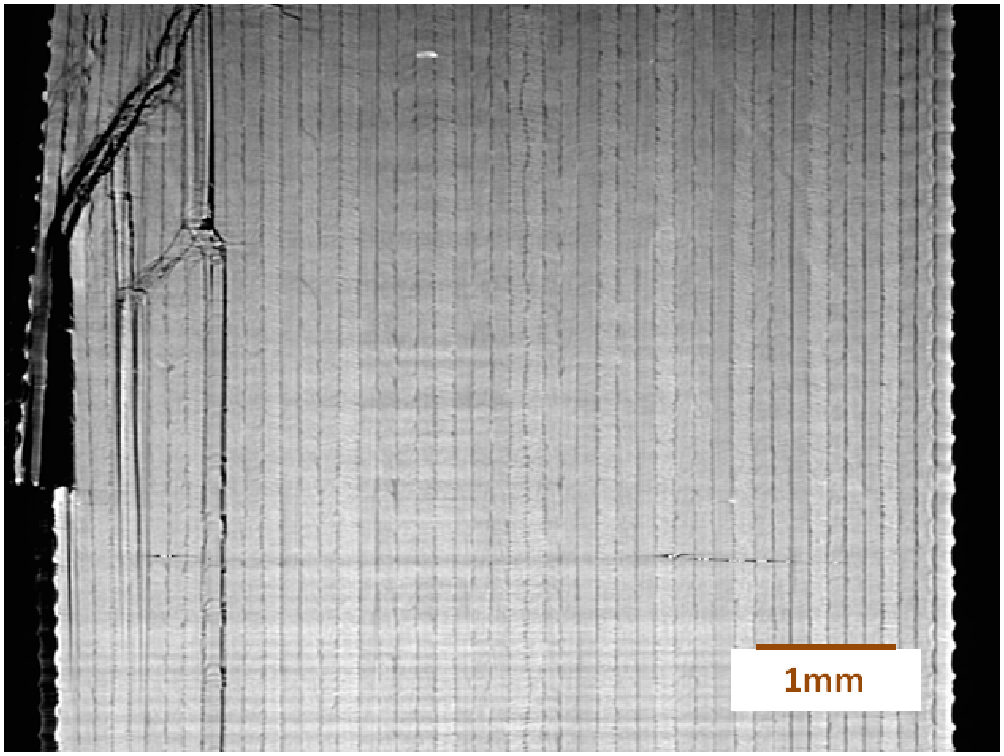


Figure 4.12 Slice from reconstruction prior to inflicting additional damage. The sample is believed to have moved due to relaxing internal stresses during the scan, causing the double-image artifact on the left side of the image while clearly imaging the primarily-undamaged plies on the right side of the image.

To test this stress-relaxation hypothesis, the sample was scanned twice after the second application of damage-inducing loads in the MTS machine. First, the sample was scanned immediately after being removed from the MTS machine, as was done after the first round. The double-imaging artifact occurs again, and is shown in figure 4.13.

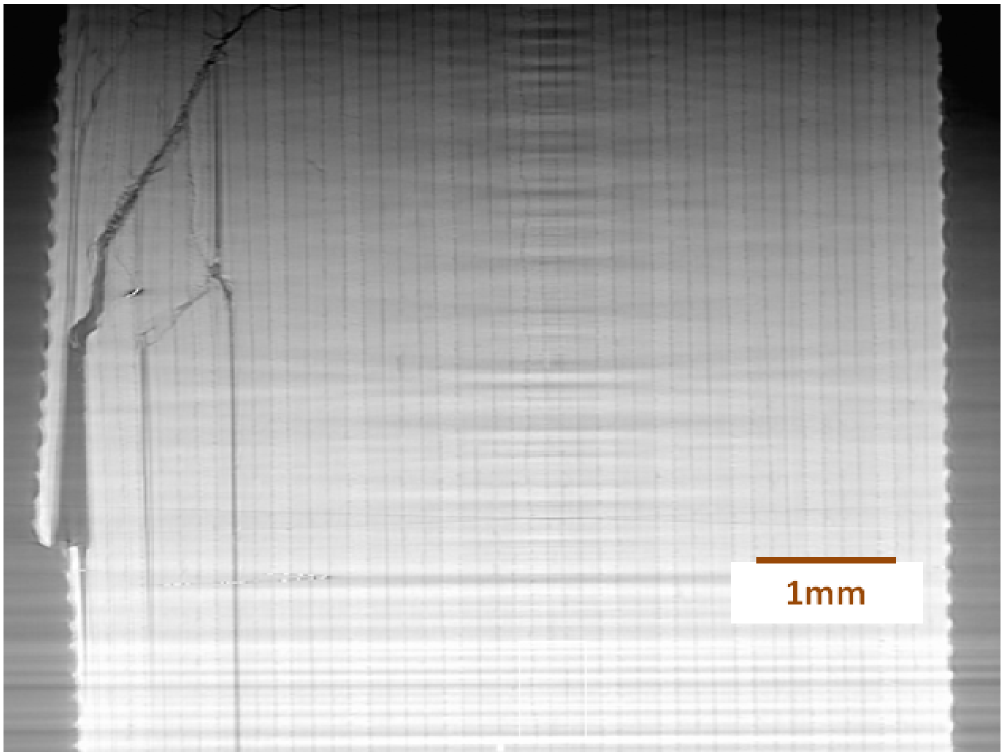


Figure 4.13 Slice from reconstruction in which the sample moved during the scan. The motion is evident by the double-imaging of the features on the right side of the image.

The sample was re-scanned the next day after it had been allowed to rest for over 24 hours. After resting, the double-imaging artifact is gone, as seen in figure 4.14.

These results indicate that further studies must either employ a fixture which prevents relaxation of the sample or allow samples to relax prior to performing CT scans.

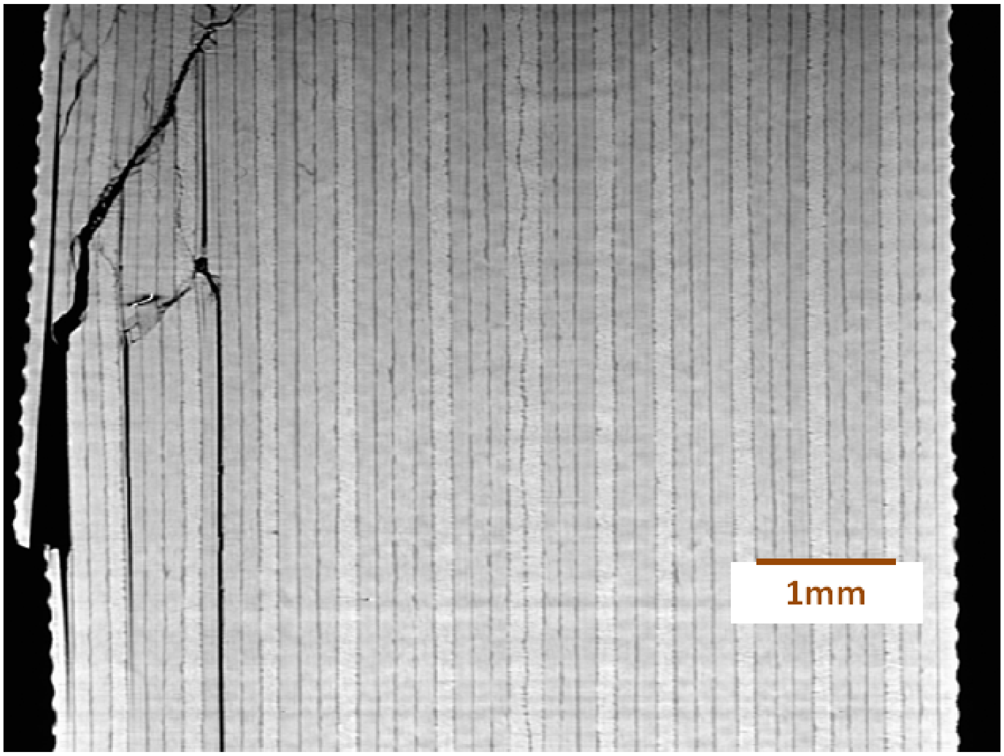


Figure 4.14 Slice from reconstruction in which the sample remained steady during the scan. Note that the double-imaging present in figure 4.13 is no longer present.

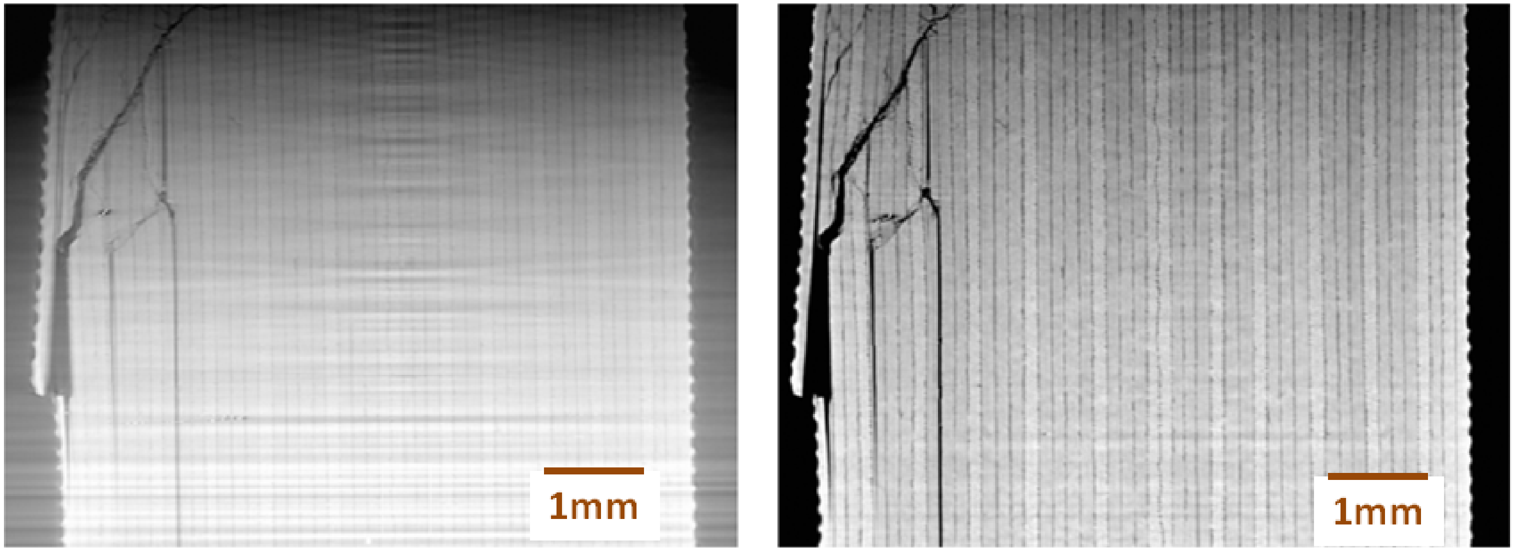


Figure 4.15 Side-by-side comparison of figures 4.13 and 4.14. The clarity of the image on the right shows the importance of allowing the sample to relax prior to performing the CT scan so that no uncontrolled motion occurs during the scan.

4.2.2 Inflicting Damage

In order to study the growth of damage structures within the sample, an additional session of loading was performed using the MTS machine. As before, the sample was monotonically deformed in a 4-point bend apparatus mounted in the CNDE MTS machine until two bursts of ultrasonic emissions were observed. The MTS force and ultrasonic emission energy data are plotted in figure 4.16.

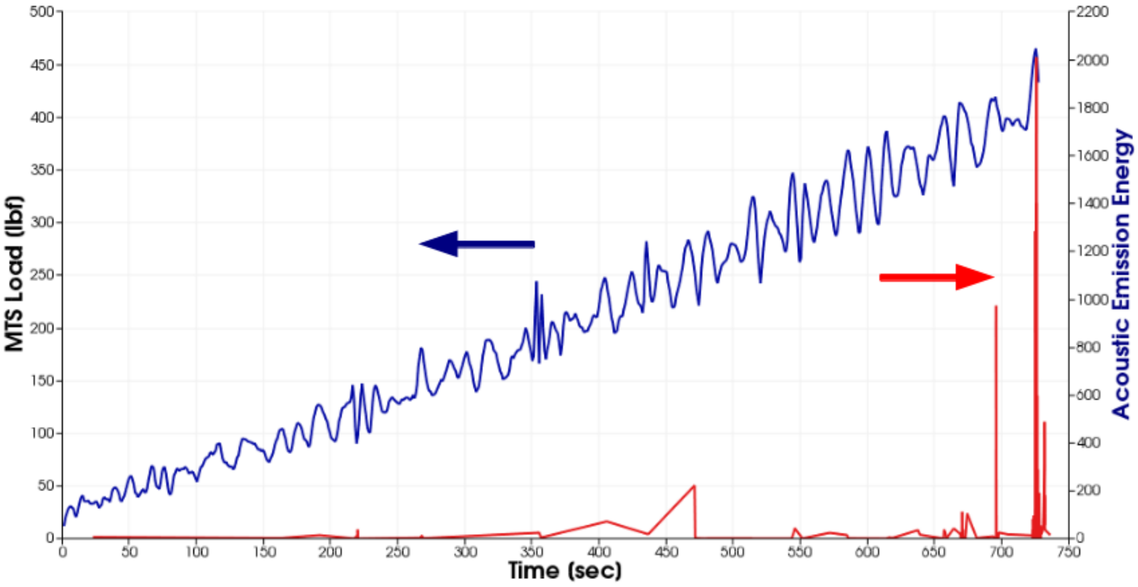


Figure 4.16 MTS and Acoustic Emission data from first round of damage-induction. Applied load by the MTS is in blue and is read off the left axis. Acoustic energy measured by the Acoustic Emission probe is in red and read off the right axis.

As noted above, it was after this scan that the stress relaxation behavior was realized. Figures 4.17 and 4.18 were generated from the CT scan after the sample had rested for 24 hours.

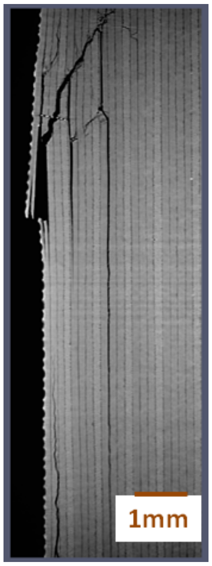


Figure 4.17 Damage after additional damage had been inflicted.

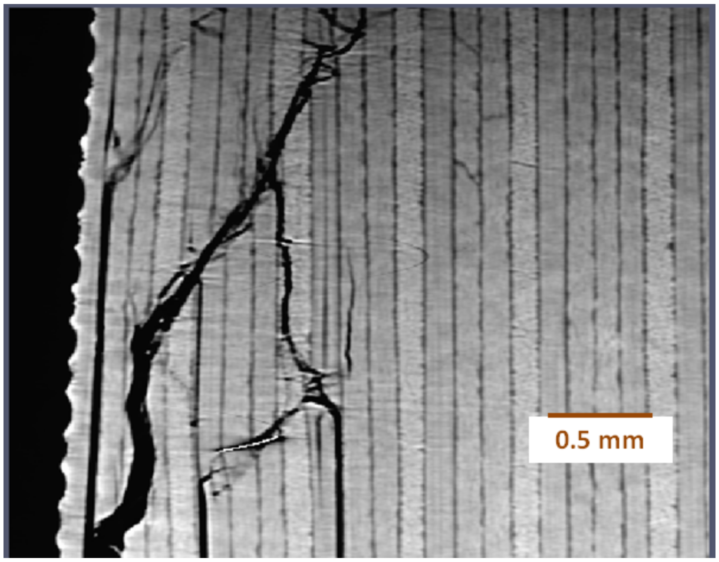


Figure 4.18 Detailed view of the upper region of damage shown in figure 4.17.

4.2.3 Observations

Despite the artifacts introduced due to stress-relaxation within the sample, it is still possible to observe changes in the structure of the damage within the sample. The comparison of before and after additional damage was inflicted can be seen in figures 4.19 and 4.20, with areas of interest identified by circled regions. Due to the artifacts present in the initial scan it is difficult to reliably identify changes in the damage structure.

When taking a closer look at the final damage structure there are several interesting features which can be seen in figures 4.21 - 4.24. Several damage behaviors are present, including delaminations between plies, cracks running within plies for significant distances, microcacking between fibers within a ply, and fiber breaks running transverse to the fiber direction within a ply.

The implication of achieving such detailed reconstructions is the realization that prognosis damage evolution models must consider all of these damage behaviors. The imaging capability demonstrated here will be of great service to provide inputs and verify the predictive power of damage evolution models. Many of the features seen in the following figures would be difficult to reliably detect using traditional microscopy methods due to the mechanical preparation of the material surface affecting the internal loads and damage structures. If the sample were to be tediously sectioned and imaged slice-by-slice by microscopy, it would be impossible to study the evolution of damage structures due to the destruction of the sample while acquiring the volumetric data.

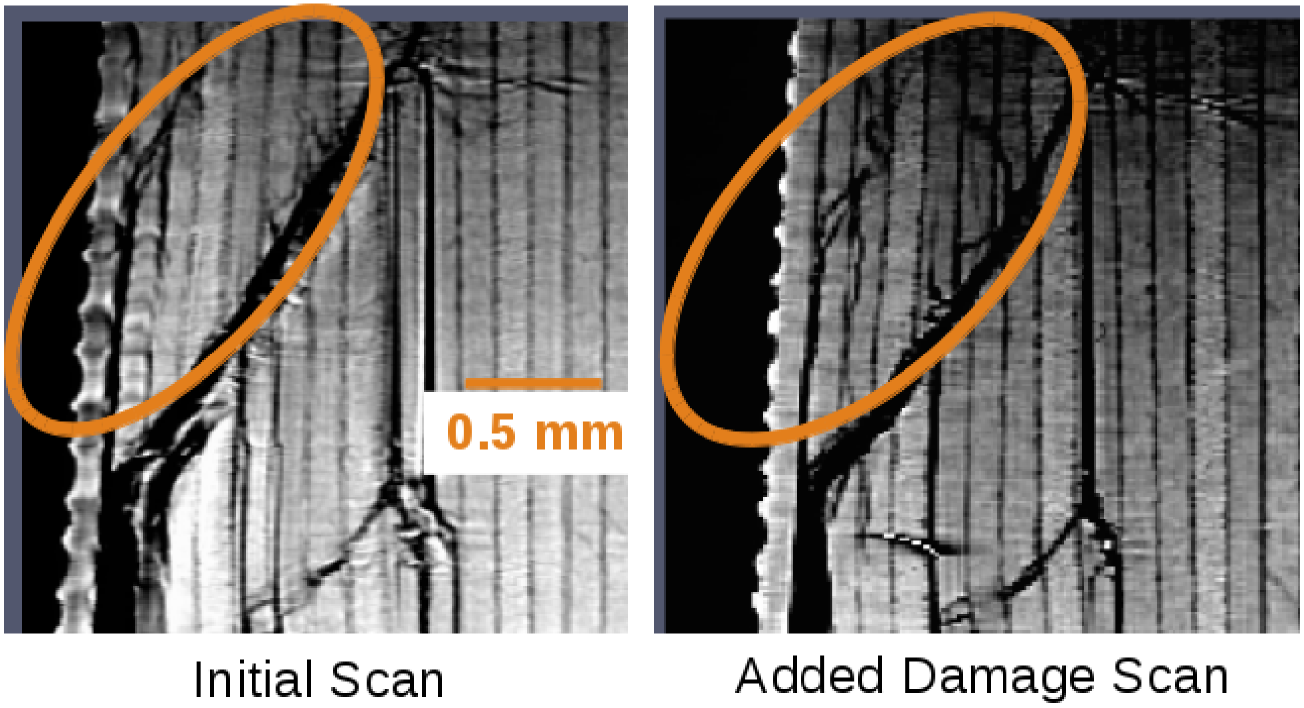
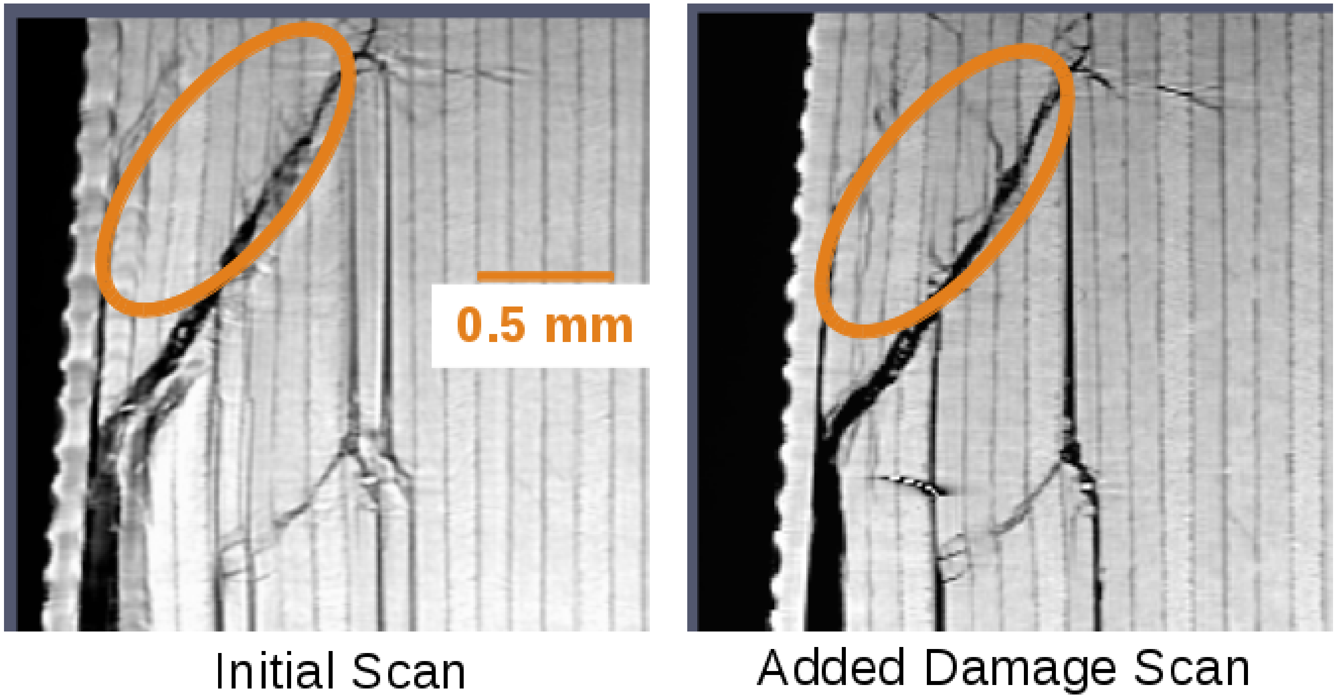


Figure 4.19 Comparison of CT reconstructions before and after damage infliction. Although the image on the left contains double-image artifacts due to sample motion during the scan, it is possible to identify changes in the damage structure, particularly in the circled region. There are two significant observations to be made in this figure. First, the image on the left demonstrates the need for sample relaxation prior to the CT scan, as was shown in figure 4.15. Second, it is possible to see an increased quantity of damage in the image on the right.



Initial Scan

Added Damage Scan

Figure 4.20 Additional comparison of CT reconstructions before and after damage infliction. Although the image on the left contains double-image artifacts due to sample motion during the scan, it is possible to identify changes in the damage structure, particularly in the circled region.

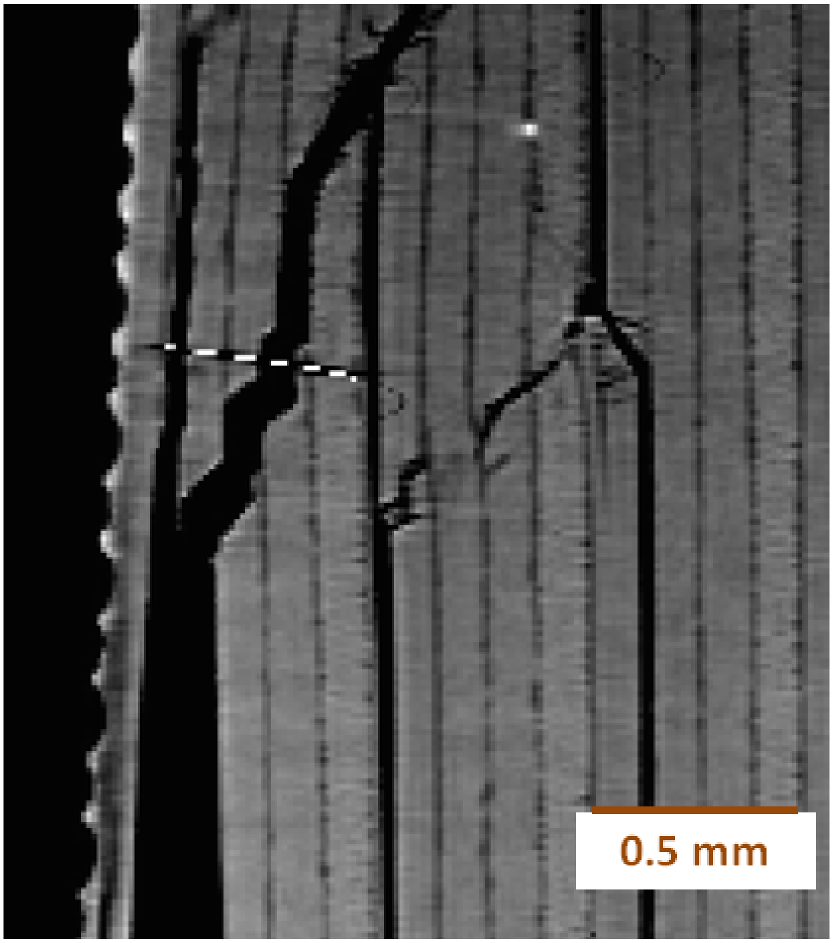


Figure 4.21 Features of interest 1. Several features can be seen, including large delaminations between plies, as well as microcracks which run along the length of the play, rather than across, forming within the plies themselves. The dashed line spanning the large crack is a reconstruction artifact.

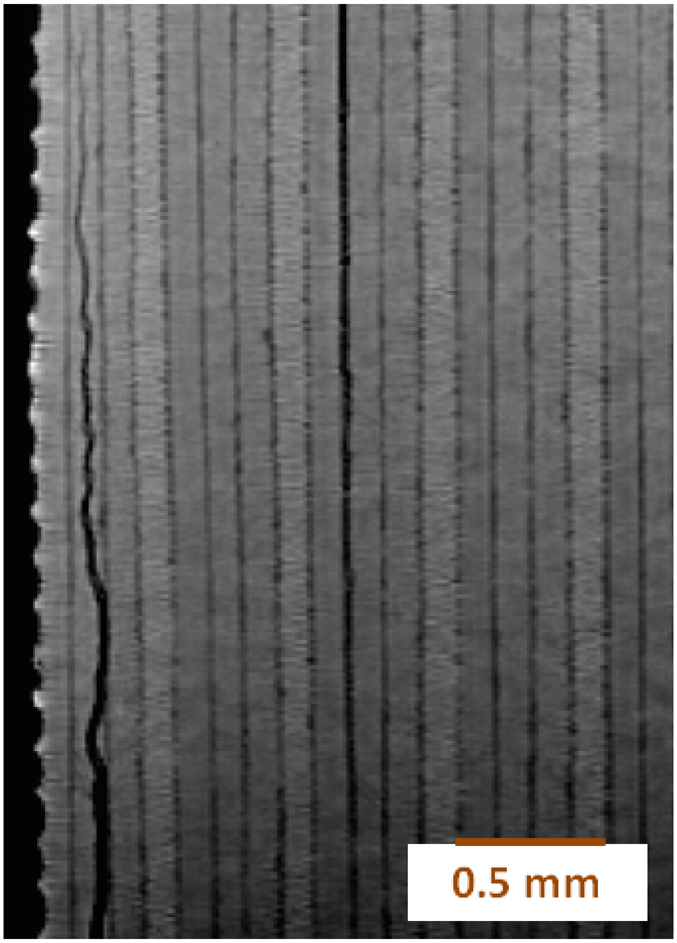


Figure 4.22 Features of interest 2. The two delaminations exhibit different structural behavior. One follows the inter-ply boundary while the other spends much of its length within a ply, indicating there are different damage-growth mechanisms at work. The delalamination/crack on the left, near the outer surface of the sample, initated at a roller contact point below the region shown in the figure. The delamination on the right is a continuation of the right-most delamination seen in figure 4.21.

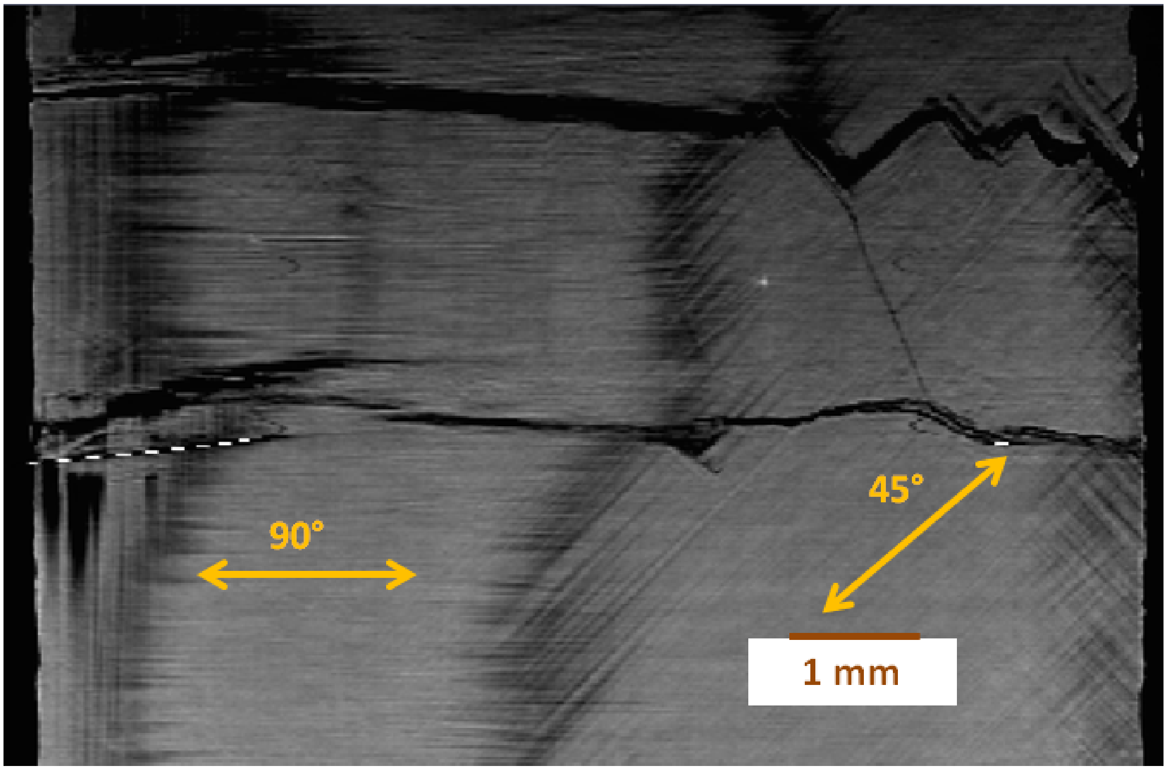


Figure 4.23 Features of interest 3. On the right side of the image, a small crack running across the ply fibers is seen to connect two much-larger cracks. The larger cracks themselves demonstrate failure along the fiber direction as well as across the fibers. Multiple fiber orientations are seen because the reconstruction grid is not aligned with the plies. The dashed line is a reconstruction artifact.

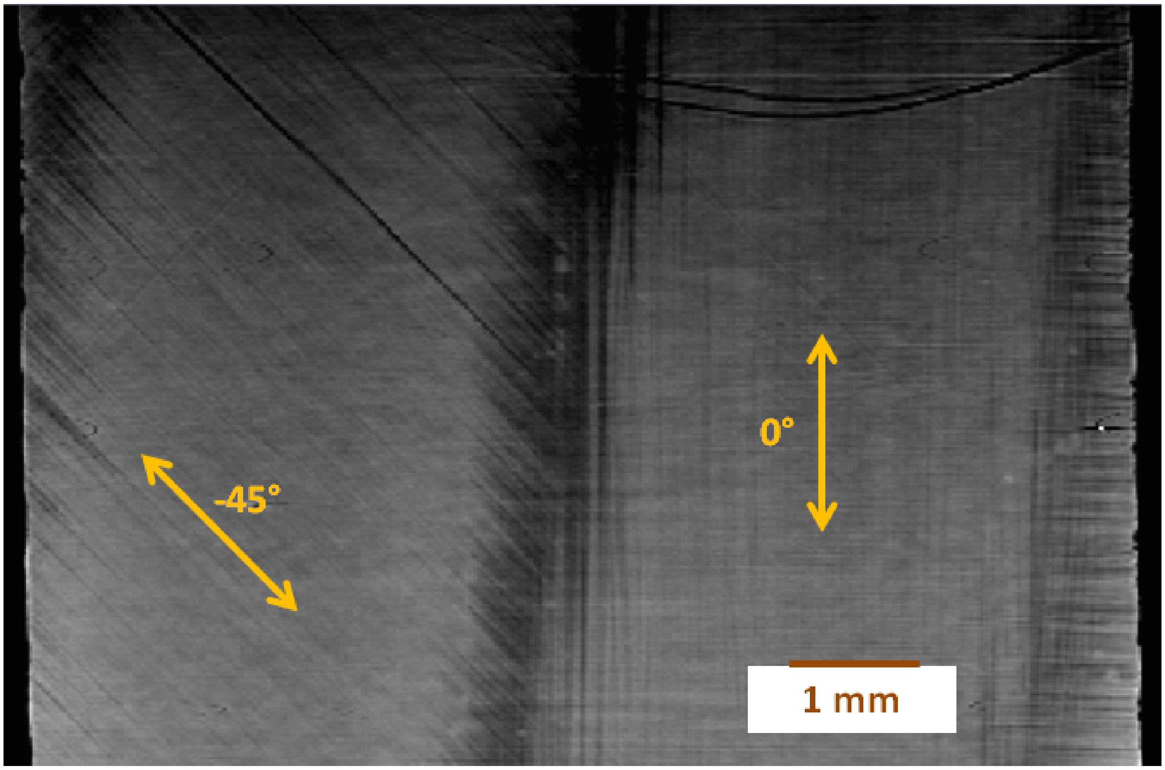


Figure 4.24 Features of interest 4. Diagonal-running microcracks can be seen in the -45-degree ply on the left half of the image. The ply in the right half of the image contains faint cracks running across the fiber direction as well as larger cracks running transverse to the fiber direction near the top of the image. Multiple fiber orientations are seen because the reconstruction grid is not aligned with the plies.

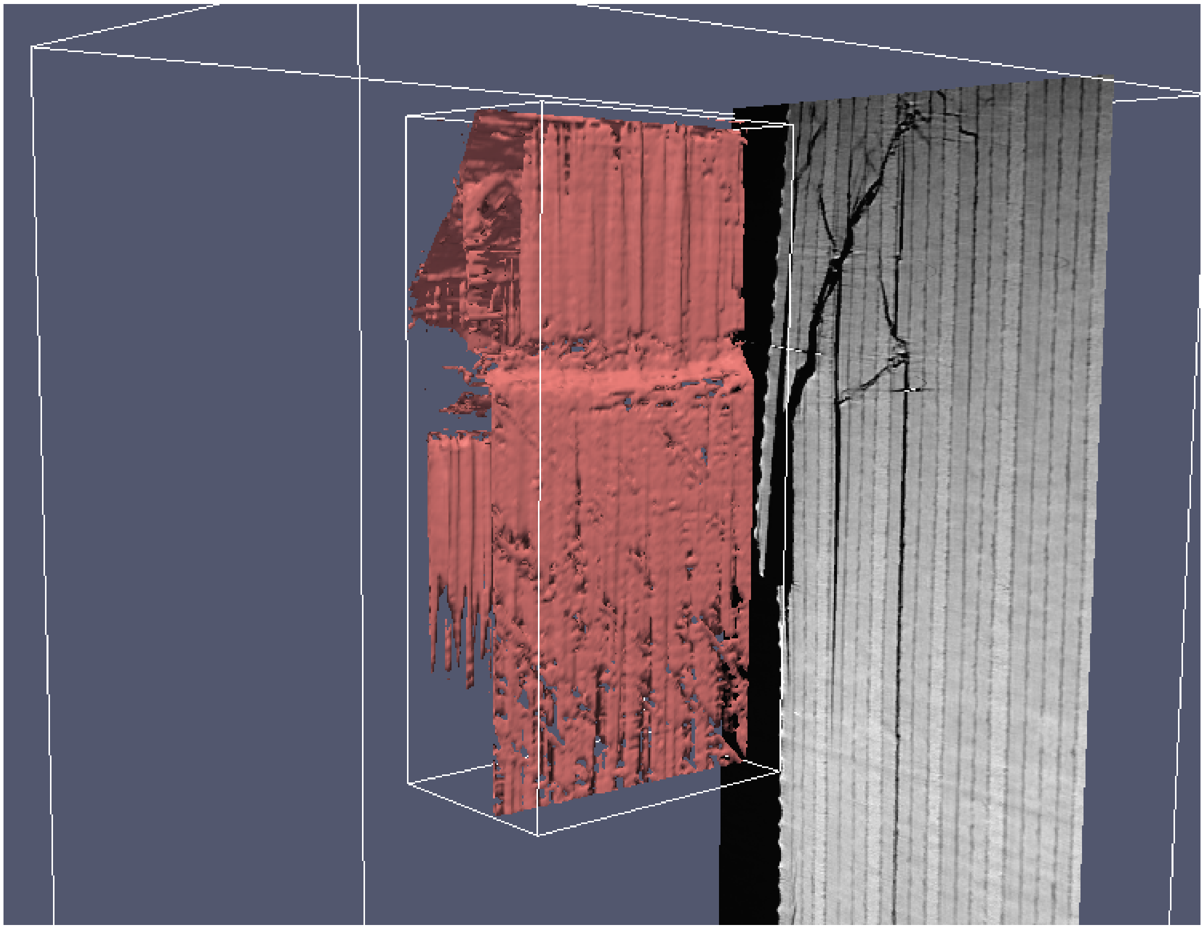


Figure 4.25 Volume rendering of damage surface. The reconstruction algorithm described in chapter 2 produces sufficient contrast to identify the damage surface using a simple threshold.

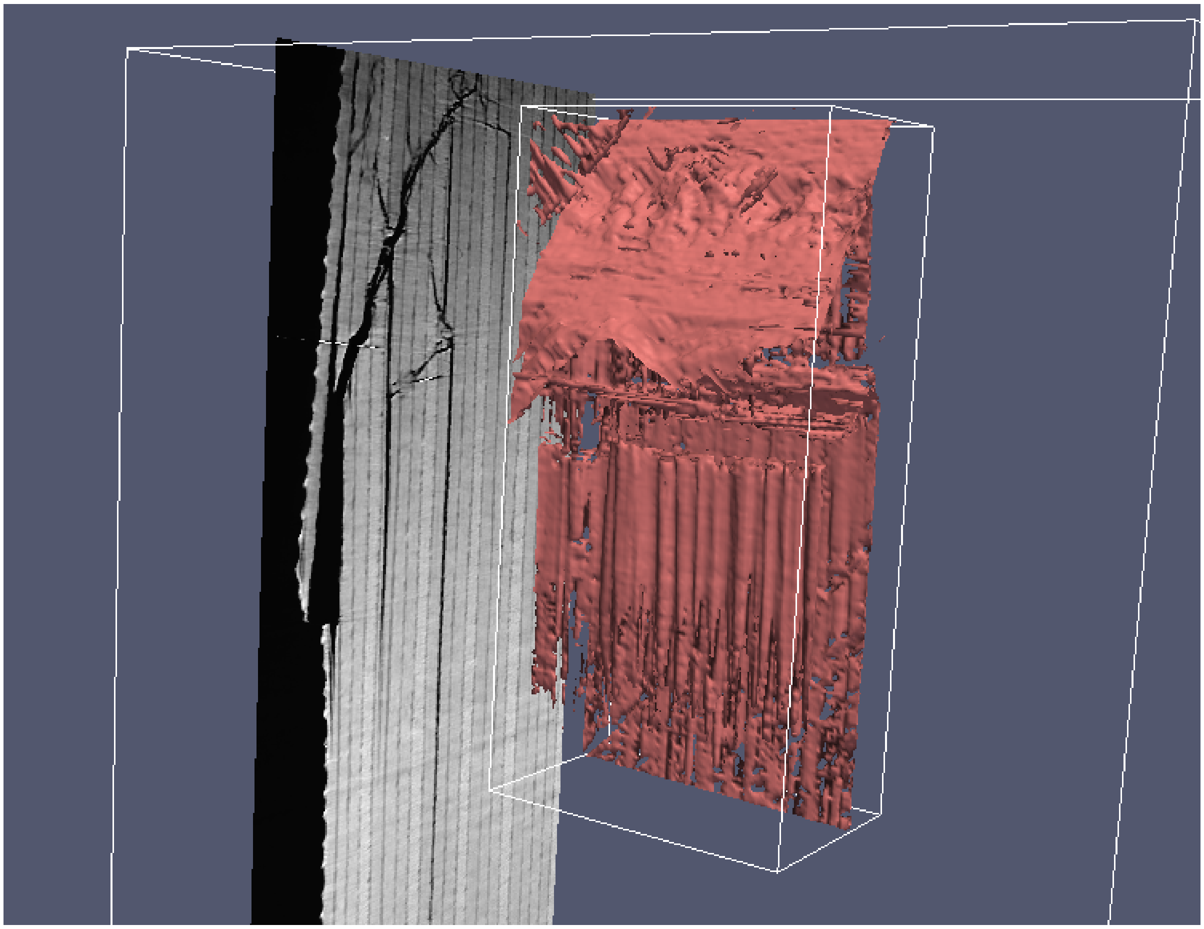


Figure 4.26 Alternate view of volume rendering of damage surface.

CHAPTER 5. APPLICATION TO ADDITIONAL GRANULAR MATERIALS

The capability of the automated defect detection routines described in section 3.1.3 will be demonstrated using two separate datasets. These datasets demonstrate two sources of low-contrast features: attenuation similarity between low-density foam and internal air-filled porosity, and small features, such as cracks and micro-porosity, in a geologic core sample.

5.1 Low-Density Foam

First, consider a piece of low-density foam, shown in figure 5.1. This sample will be used to demonstrate each step in the post-processing analysis.

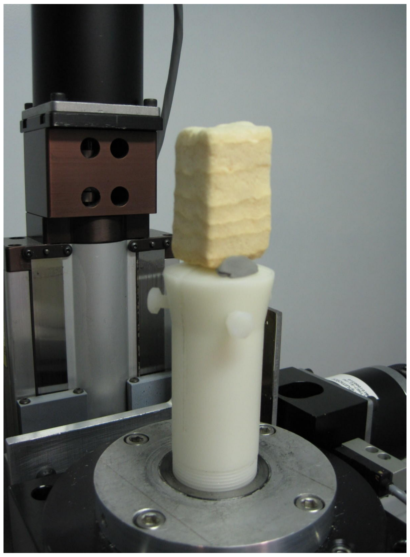


Figure 5.1 Photograph of low-density foam.

A CT slice of this sample was shown in figure 2.12 and is shown here in figure 5.2.

The first step in the process is removing any long-length trends in the dataset, and producing

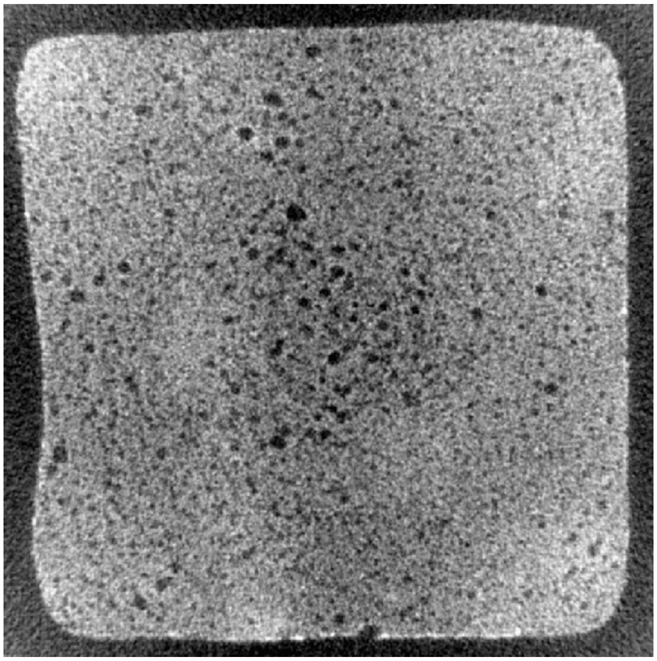


Figure 5.2 CT slice of low-density foam.

what is referred to as a “flat” dataset. Despite being a very low-density and low-absorbing material, the reconstruction of the foam still contains underlying trends which will adversely affect the statistical analysis algorithms.

Next, two subsets of the data are defined. First, a region is defined which is wholly contained within the sample in which the defect-detection analysis will be performed. The analysis is contained within the sample in order to avoid spurious edge effects which are produced by the data-flattening operation at the sample edges.

Second, a region in the sample which describes the background noise is defined. In this particular material there is no region entirely free of porosity. As a result, the best that can be done is to select a region with no major porosity and average-out the effects of the small pores by defining a large-enough reference region.

These regions are shown in figure 5.4, with the analysis boundary shown in green, the reference boundary shown in red, and the test window is shown in blue. Note that the test window is swept through the data during the analysis while the analysis and reference boundaries remain stationary.

Once the analysis and reference boundaries are defined, the statistical analysis algorithms

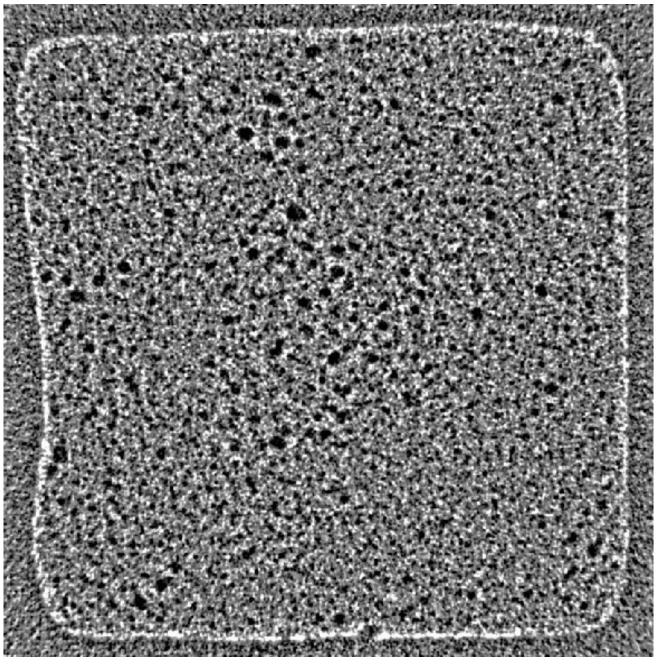


Figure 5.3 Flattened CT slice of low-density foam.

can be applied. In this example the Kolmogorov-Smirnov test using a signed distance metric, as described in section 3.1.3.2, was chosen and used a 5x5 test window. The result is then thresholded to only show regions with significantly-large distance magnitudes. By setting the threshold range to [-1, -0.5], the porosity within the foam is readily identified, as shown in figure 5.5. A more-detailed view of the lower-right corner is shown in figure 5.6.

We can see in figure 5.6 that very small pores are left unflagged. This is an effect of the test region's footprint. As the size of the test region grows, larger features are required to meet the criteria of the statistical test. Smaller test regions are more sensitive to small, subtle features but are also more sensitive to random noise. The chosen size of 5x5 has been found to work well in most instances, although it can be seen in figure 5.6 that there is a noticeable detectibility limit.

This difficulty with small features is shown in figures 5.7 and 5.8.

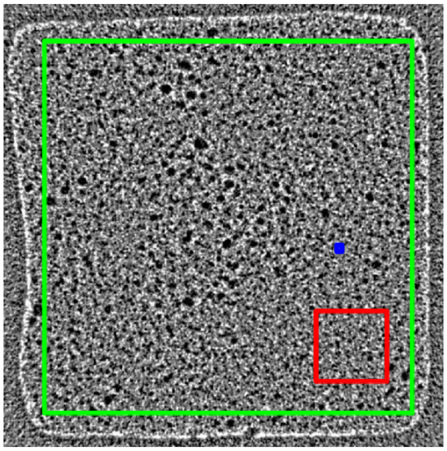


Figure 5.4 Flattened CT slice of low-density foam with analysis regions shown. The defect-detection routine is confined to the green box in order to avoid edge-effects at the outer edges of the sample. The reference, background noise is defined by the red box. The test window is shown in blue, and is swept through the data during the analysis process.

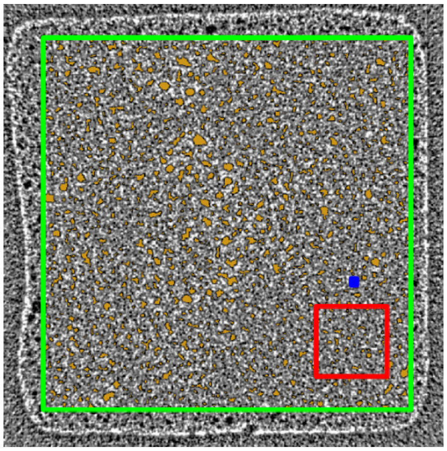


Figure 5.5 Detected porosity in low-density foam. This result was obtained using the Kolmogorov-Smirnov statistical test with the signed distance metric (described in section 3.1.3.2). Values between [-1, -0.5] are displayed in orange.

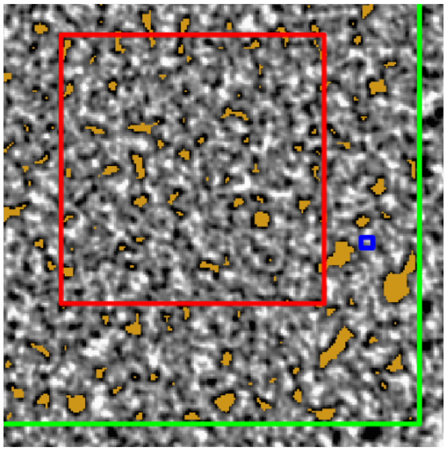


Figure 5.6 Detected porosity in low-density foam, detailed view. This is the lower-right corner of figure 5.5. Note that porosity within the reference region is readily identified, despite being included as part of the definition of the background.

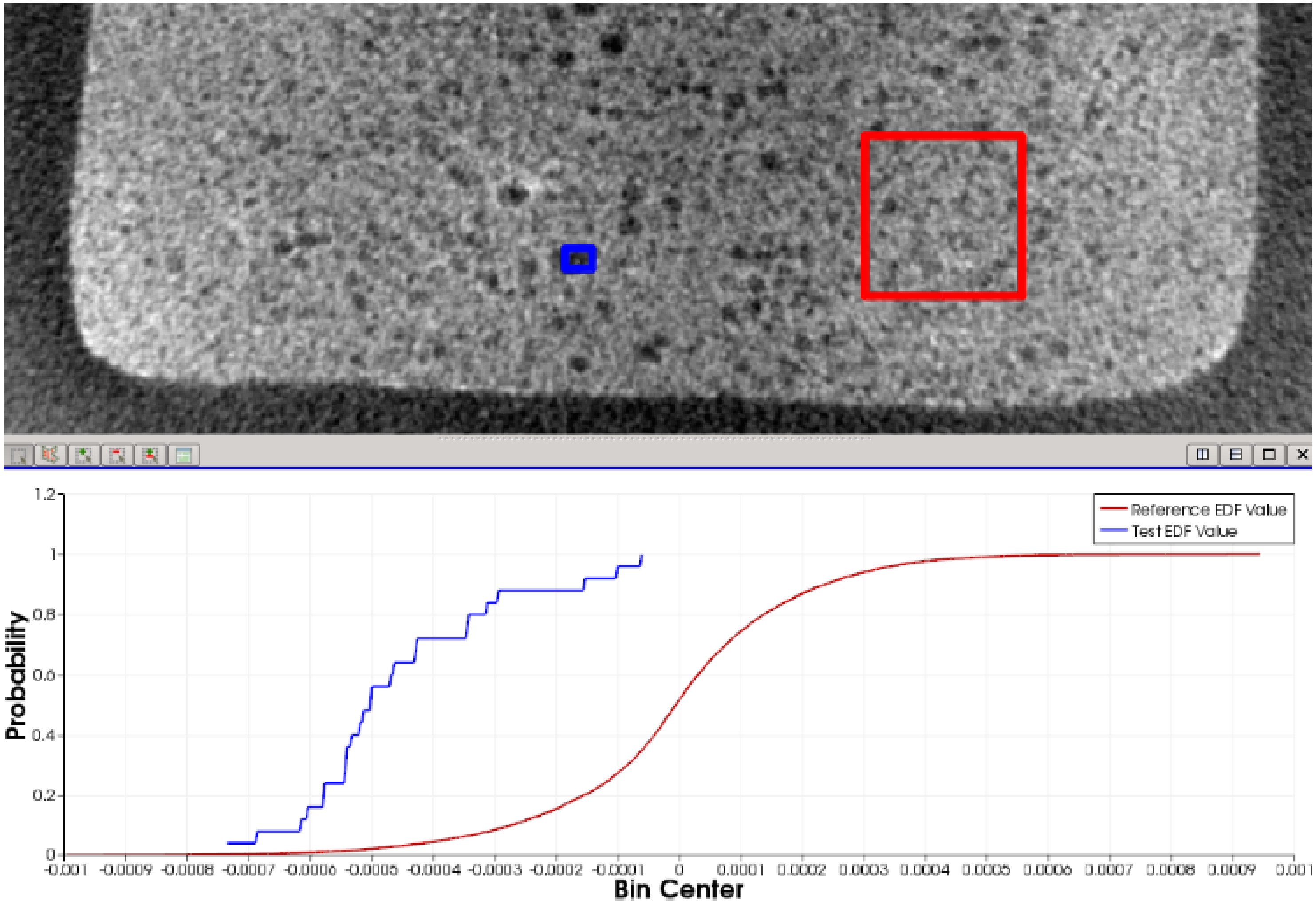


Figure 5.7 Volumetric effect of test area on large feature. Note that the empirical distributions are significantly different, producing a signed distance metric of -0.79. The negative value indicates that the feature is porosity rather than a high density inclusion.

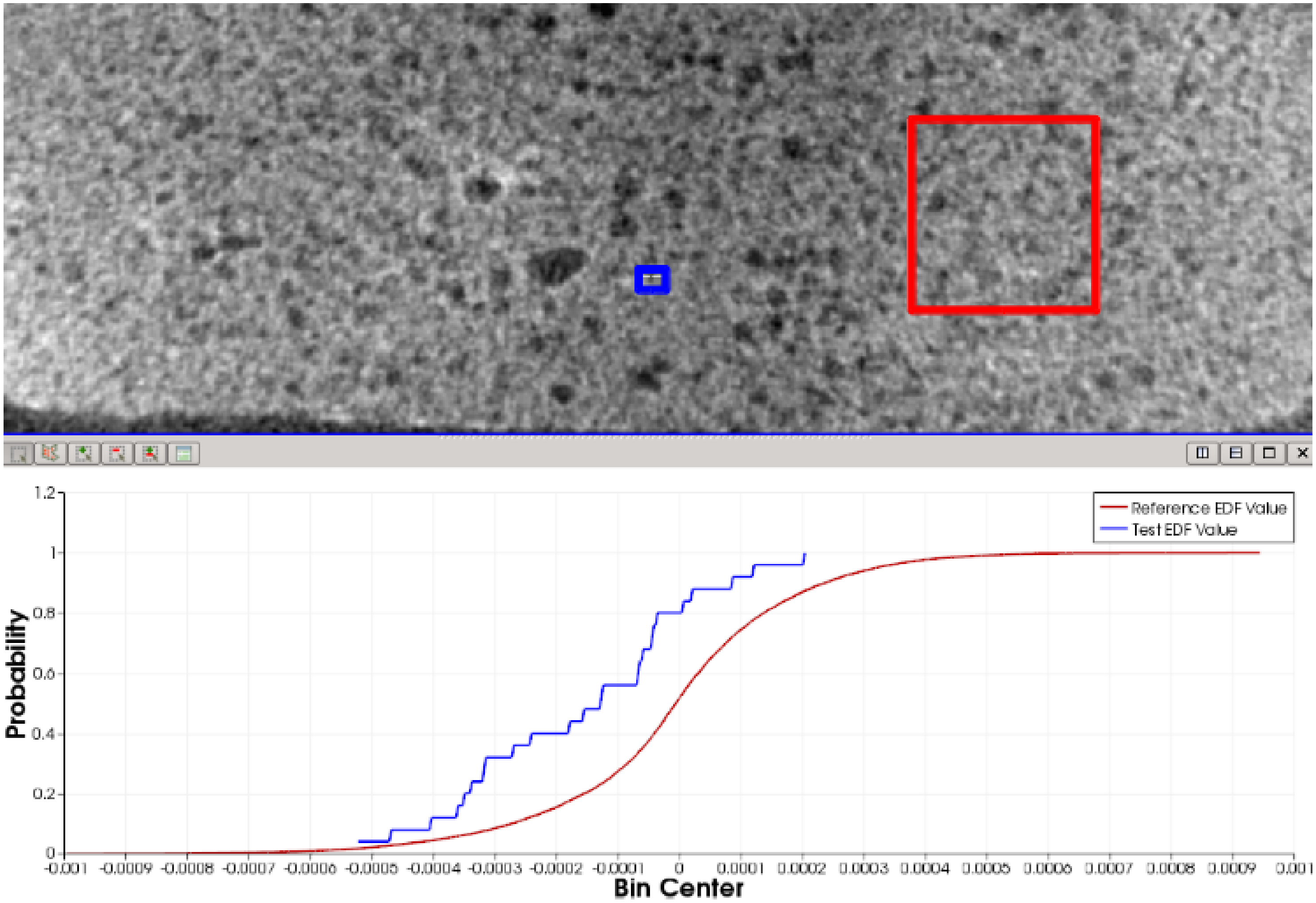


Figure 5.8 Volumetric effect of test area on small feature. Note that although the empirical distributions are obviously different, the signed distance metric is only -0.38. The negative value indicates that the feature is porosity rather than a high density inclusion. The Kolmogorov-Smirnov algorithm is not sensitive to the fact that the empirical distributions do not cross, which suggests to a human operator that there is likely a feature there despite the fact that the distance between the two distributions remains modest.

5.2 Dolomite

As a final example, the above-described process was repeated for a dolomite rock core, and the result is shown in figure 5.9.

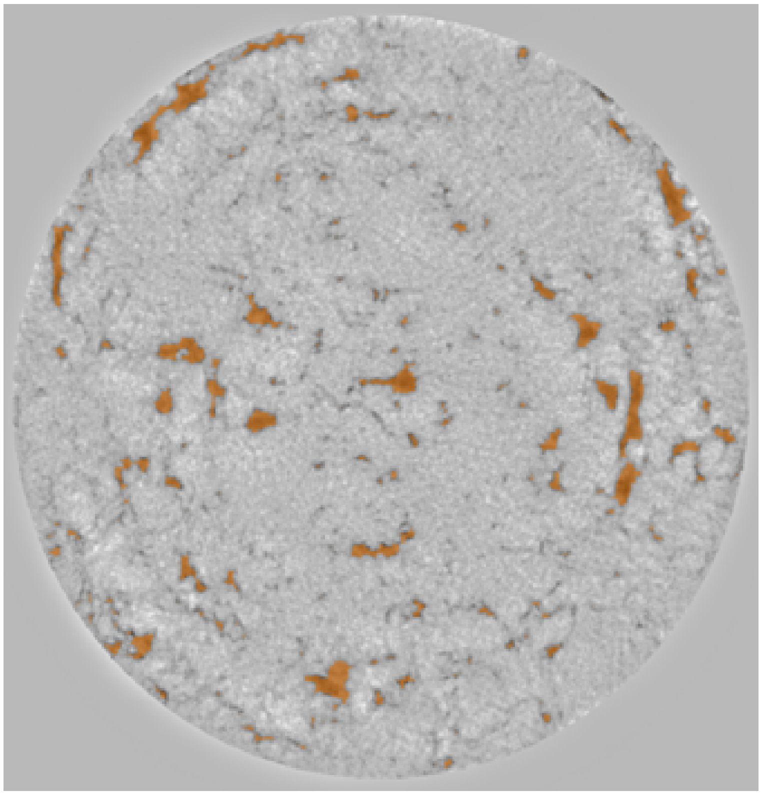


Figure 5.9 Detected defects in a dolomite rock core.

The large-scale porosity was readily identified, and many of the smaller pores were also successfully found. This is due to the selected test window being preferential towards larger, volumetric features, such as porosity, rather than small, linear indications. Such behavior was also observed with the low-density foam and illustrated in figures 5.7 and 5.8. The challenge in this dataset is capturing the subtle crack network, and much of the cracks were left undetected for the chosen threshold range.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Developing robust structural health monitoring capabilities and prognosis models requires a detailed understanding of the damage state within a material. When those capabilities and models are intended to work with early-stage damage, well-before the damage has began to grow according to well-known models such as the Paris Law, a means of characterizing damage on the granular-scale of the material is required. This length-scale is material dependent and can vary from fractions of microns in ceramics, to single microns in fiber-reinforced composites, to thousands of microns in soil and concrete. Historically, characterizing fiber-reinforced composites on this scale has required optical microscopy, which incurred significant time and material costs. Further, optical microscopy requires destructive polishing and, in the case of volumetric studies, surface removal which prevents its use in damage-evolution studies.

Despite significant advances over the past decade in the ability to use CT to image fiber-scale damage structures in fiber-reinforced composites, further improvements were needed to enable the study of early-stage damage and the evolution of that damage as the material was placed under load. Resolutions needed to be pushed to the point of being able to image the fiber-scale damage structures while using a typical laboratory CT system and without requiring the use of additional contrast agents.

By developing a new reconstruction algorithm,designed for execution on graphical processing units (GPUs) and involving an improved noise-suppression filter which significantly improves image contrast, voxel sizes on the order of 2 μm have been achieved, a factor of 5 improvement over current capabilities, while using a typical cone-beam laboratory CT system. In addition to the increased spatial resolution, the improved image contrast enables the detection of small, subtle, low-contrast signals. Early-stage damage begins as small-scale features within the material, and the resulting NDE signals are subtle as a result.

This imaging capability supplies the missing tool which enables accurate characterization of early-stage damage within fiber-reinforced composites. The ability to quantify the internal granule-scale structure of a material is a critical step for developing and testing damage evolution models. Such models are themselves a key component of prognosis. Further, structural health monitoring attempts to use low-cost sensors to characterize the current state of a material. Such sensors often produce ambiguous signals, such as the lack of location information using acoustic emission to guide damage-induction in materials studies, and developing the capability to decipher these ambiguous signals requires the ability to characterize granule-scale structure within the material.

In addition to improving the quality of CT reconstructions, it was also necessary to develop physics-based image processing routines which provide robust defect detection capabilities while using a minimum of operator intervention. Such algorithms are a necessity when working with high-resolution CT due to the massive quantities of data which are produced. A typical CT scan shown in this dissertation involved 40 gigabytes of projection images and 15-20 gigabytes of reconstruction volume. Further, it was often necessary to perform multiple reconstructions for a particular collection of input data in order to realize the full resolution which is now possible at CNDE.

The improved imaging capability coupled with the robust post-processing algorithms form an improved experimental tool which is already being used for additional material characterization studies at CNDE. With this new tool it will be possible to gain significant insights into the key factors involved with the determination of material allowables, structural health, and prognosis.

6.2 Future Work

6.2.1 Improved Post-Processing Performance with Complex Materials

The post-processing routines described in chapter 3 work very well with volumetric features. Their performance with linear features, such as cracks, suffers due to the smaller spatial extent of the features as well as the reduced magnitude of the features' signatures. Extending the post-processing algorithms to improve their sensitivity to crack-like features would be of great use when searching for the first signs of damage within a CT reconstruction volume.

An additional challenge faced with the current post-processing routines is their performance with complex, highly-structured materials, such as fiber reinforced composites. The ideal analysis algorithm would identify only the anomolous features and ignore the background structure of the sample itself, but the implementation of such logic into the algorithm is a complex and daunting task.

6.2.2 Rigorous Quantification of Acoustic Emission Capability as a Guide for Damage Induction

The efforts described in chapter 4 involved a very basic use of the acoustic emission (AE) inspection equipment. The principle of using a technique such as AE to govern the process by which damage is incrementally induced in a sample is sound, but there are several issues which must be resolved before AE can be reliably used to guide a detailed materials study.

1. First and foremost, effort must be spent to determine the optimum acquisition and filtering settings for the AE equipment itself. It is reasonable to expect that these settings will be material dependent, so a methodology for determining the appropriate settings must be developed.

2. Also, work must be done to verify that the measurements made by the AE equipment are actually quantifying the sample rather than the MTS machine or other test apparatus. The noticeable difference between the cyclic MTS force and delta-like AE energy suggests that this is case, but verification is needed.

3. It is reasonable to expect that different internal failure mechanisms can produce unique AE signatures. If it were possible to identify these unique signatures AE would not only be able to guide the damage-induction by informing the user *when* damage has occured, but also *what type* of damage occurred.

4. The AE equipment at CNDE has many methods of quantifying the measured signal, such as amplitude, energy, duration, and so on. The work in chapter 4 looked at the energy of the emissions since load-shedding events were desired and the load would necessarily be discharged into some form of energy. Other quantification methods may prove more appropriate or effective, both in terms of when damage has been induced and how much damage has been induced. Understanding the relationship between AE measurements and the quantity of induced damage would be a signficant help when performing future materials studies.

6.2.3 Utilization of Imaging Capability in a Detailed Materials Study

The work described in chapter 4 is a preliminary glimpse into how the improved imaging and post-processing capabilities may be leveraged to perform advanced materials studies. The following methodology is proposed for future studies:

1. In order to maximize the spatial resolution of the scan, keep the samples small. With the current system at CNDE it would be best to keep the samples small enough to fit within a cylinder which has a diameter of 4 millimeters. The goal is to have the sample fill the detector's field of view when the sample is placed at the maximum-possible magnification.

2. Prior to inducing any damage, perform a detailed scan of the entire sample. Damage initiation is controlled by features far-smaller than what may be imaged with most NDE techniques. As a result, it is difficult to robustly predict the location of damage initiation and it is necessary to protect oneself against the possibility of damage appearing in an unexpected region.

3. Having already addressed the necessary tasks regarding the use of acoustic emission (AE) monitoring, use AE to guide the damage-induction process. Rather than using the histor-

ical approach of characterizing the condition of a material at a fixed temporal interval, let the measurement interval be controlled by the condition of the material itself. The preliminary work in chapter 4 has shown that techniques such as AE can nicely compliment the more-detailed analysis technqiues.

4. After the damage has been induced and it is time to characterize the new material state, either,

    - Utilize a special-purpose mount in the CT scanner which will prevent the sample from deforming as stresses attempt to relax during the CT scan. Introducing additional materials into the radiation beam, however, brings a host of additional challenges so this approach must be chosen with extreme care.
    - Allow the sample to rest for a period of time so that any stress relaxation and deformation which will occur can happen outside the CT scan. Even miniscule movements during high-resolution CT scans can be extremely detrimental to the quality of the reconstruction volume. Due to its simplicity, this “let it rest” approach is the recommended method, provided the features of interest within the material can be imaged without an external load to maintain a particular orientation or condition.

5. Prior to actually performing a post-damage-induction CT scan, acquire a series of flat radiographs to identify the newly-damaged regions. This is simply a matter of practicality due to the time required to acquire a CT scan and the hard drive space required to store all the data.

6. Perform high-resolution CT scan(s) of all regions which have been identified as containing damage. This includes regions which previously had damage, so the growth and evolution can be studied, as well as newly-damaged regions, so the initiation process may be observed.

7. After performing the needed CT scans and verifying the quality of the data, induce more damage and repeat the process.

APPENDIX A. SURVEY OF CT RECONSTRUCTION METHODS

Inversion-Based Reconstruction

The inversion methods perform the reconstruction by calculating the inverse of the Radon transform, often using the relationship between the Radon and Fourier transforms to accomplish the task. As shown by Herman and Lung (1980), the inversion methods are derived and described over all real numbers (i.e., continuous functions and data). After all derivations are complete, discretization is then considered to allow numerical implementation.

Direct-Inversion Methods

As the name suggests, direct-inversion methods attempt to perform the reconstruction of the sample by directly inverting the projection operation which occurred during the data collection. This involves inverting the Radon transform.

Herman (2009) shows that the inverse Radon transform can be calculated using a series of three operators:

$$\mathcal{R}^{-1} = -\frac{1}{2\pi}\mathcal{B}\left(\mathcal{H}_Y\left(\mathcal{D}_Y\left(p\right)\right)\right) \tag{A.1}$$

where

- p is the projection data expressed as a function of two variables, $p(l, \theta)$.
- $\mathcal{D}_Y$ produces a new function $q(l, \theta) = \frac{\partial p}{\partial l}$.
- $\mathcal{H}_Y$ produces a new function t, which is the Hilbert transform of q with respect to l.
- $\mathcal{B}$ is the backprojection operation applied to function t.
- $-\frac{1}{2\pi}$ is a normalization constant.

Herman notes that this process assumes exact values of $p(l, \theta)$ are known for all l and θ and that all mathematical operations can be performed precisely. Use of digital detectors and computers violates these assumptions, resulting in the algorithm output becoming an estimation of the sample rather than a direct solution (Herman (2009)).

An alternative inversion technique, called Circular Harmonic Decomposition, involves calculating the reconstruction image as a series of annular rings. The sinograms of collected data are usually analyzed as a set of individual rows, where each row in the sinogram corresponds to a row of the detector. If, instead of analyzing horizontal rows, vertical strips are considered, these vertical strips will correspond to annular rings in the image. The data contained in these rings is periodic in θ with a period of 2π (where θ is the angular orientation of the sample and it is assumed that the data were collected by rotating the sample about an axis). This technique is explained more-fully by Barrett and Swindell (1981).

Yet another alternative is to rearrange the measured data into what is called a *linogram*. A point in the sample follows a sinusoidal path in conventionally-measured data, hence the name "sinogram". A linogram remaps the sinogram data so that all rays which pass through a fixed point now correspond to a straight line. Use of special inspection geometry, such as linear computed tomography, can also cause all rays which pass through a fixed point to lie in a straight line in the measured data. Edholm and Herman (1987) have shown that linograms produce the same result as filtered backprojection and claim, along with Herman (2009), that reconstruction based on linograms can perform reconstructions faster than filtered backprojection ($O(N^2 logN)$ vs. $O(N^3)$. Gao et al. (2006, 2008) applied linogram-based reconstruction to the case of limited-angle tomography.

Algorithms based on direct Fourier inversion do not appear in the transmission tomography literature as often as other methods. They are, however, used more often with magnetic resonance imaging (Natterer and Ritman (2002)).

Indirect-Inversion Methods

Filtered backprojection methods are very popular and can also be referred-to as "convolution methods". These methods combine the derivative and Hilbert transform operations into a single convolution operation. Details showing the correspondance between derivation/Hilbert transform and convolution are provided by Herman (2009).

Implementation of FBP methods often utilize the Fourier Transform to calculation the convolution in an efficient manner. Barrett and Swindell (1981) show that backprojecting 1-dimensional filtered data and performing a 2-dimensional filtering operation on the backprojection result (backprojection done without prior filtering in this case) are mathematically equivalent. This process has been shown to be correct for both parallel-beam and fan-beam geometries.

For simple inspection geometries, the backprojection operation may be performed using a process known as "shift and add" (Dobbins and Godfrey (2003)). This method is easily derived for both parallel-beam and fan-beam geometry, and can be implemented in a very computationally efficient manner. Its greatest limitation stems from its use with limited data situations and the resulting artifacts. These artifacts are also present when traditional backprojection is applied to limited data.

Series-Expansion-Based Reconstruction

Herman (2009) classifies everything that is not an inversion of the Radon transform as a "series expansion" method. These methods begin by discretizing the reconstruction domain first and then developing algorithms which are tailored to work with a discrete domain. Conceptually, these approaches are independent of the choice of basis functions for the image, although achieved results are dependent upon the choice of basis functions.

Choice of Basis Functions

When discretizing the reconstruction calculations a set of basis functions must be chosen. As described by Herman (2009), the chosen basis functions should have several properties:

- The image which is to be reconstructed may be approximated by a linear combination of basis functions.

- The basis functions should be linearly independent so that there is a unique solution which best-approximates the correct reconstruction result.

- The basis functions should be square integrable.

A common choice of basis function is a simple piecewise-constant value for each pixel. Mathematically, if there are J pixels in the reconstruction, the basis function on the i^{th} pixel is

$$b_i(r,\phi) = \begin{cases} 1, \text{if } (r,\phi) \text{ is inside the } i^{th}\text{pixel} \\ 0, \text{otherwise} \end{cases} \tag{A.2}$$

For some algorithms this choice of basis functions can cause too much noise in the final result. An alternative choice of basis functions are the Generalized Kaiser-Bessel window functions, referred to as “blobs” in Herman (2009). Use of blobs instead of the more-traditional pixels can produce smoother reconstruction results due to their more diffuse shape.

Algebraic Methods

Algebraic methods attempt to perform the reconstruction by solving the matrix equation

$$A\mathbf{x} = \mathbf{b} \tag{A.3}$$

where $\mathbf{b}$ is the measured intensity for each pixel in each exposure, $\mathbf{x}$ is an unknown vector of linear attenuation coefficients for each voxel in the reconstruction volume, and A is a matrix of weight factors which relate the contribution of each voxel’s attenuation to the value measured on the detector.

The phrase “algebraic reconstruction techniques” is applied to all methods that begin with a formulation such as in equation A.3, regardless of the techniques used to find the solution for the unknown vector $\mathbf{x}$ (Herman (2009)). Solution by direct inversion of A is impractical for two reasons:

1. A is only square if the number of measurements (i.e., number of detector pixels multiplied by the number of exposures) equals the number of voxels. This is often not the case.

2. The number of calulations and memory required to perform Gaussian elimination and back-substitution are prohibitive for all but the smallest inspections.

As a result, research focuses on iterative solution methods based on linear algebra (e.g., method proposed by Kaczmarz (1937)), optimization theory (Herman (2009)), and statistical analysis of the x-ray physics and measurement process (Sonka and J. M. Fitzpatrick (2000)). Although these approaches are interrelated and not mutually-exclusive, they provide a useful means of categorizing algorithms.

Linear Algebraic Methods

In the literature, when the term "ART" appears it is often referring to a specific approach to solving equation A.3. Mathematicians know this approach as "Kaczmarz's method" (Kaczmarz (1937)). In this method, each element of $\mathbf{x}$ is considered separately and successive iterations to each element are made without considering the effects on other elements within the vector. This technique has been found to converge to a solution faster than other algebraic techniques (i.e., SIRT), but at a cost of greater amplification of measurement noise than the other techniques (Dobbins and Godfrey (2003)). A common method of controlling noise is to under-relax the iterative approach (Herman (2009)). Under-relaxation improves performance with respect to noise at the cost of greater numbers of iterations required.

Linear algebra approaches may also be used to de-blur a reconstruction produced using unfiltered backprojection. In the case of limited data the filter functions used by the inversion reconstruction methods introduce artifacts into the result. Several methods for de-blurring unfiltered backprojection results are described by Dobbins and Godfrey (2003). Naturally, the methods differ in their details and implementation, but they share the common trait of specifying some sort of correction function for the blurred image and then using linear algebra techniques to apply the correction function.

Optimization Methods

The algebraic system of equations defined in equation A.3 can also be solved using optimization approaches. One such approach is known as "Simultaneous Iterative Reconstruction

Technique" or SIRT (Dobbins and Godfrey (2003); Herman (2009)). In SIRT, updates to all elements of vector **x** are made simultaneously rather than individually. By considering all updates for a given iteration simultaneously the reconstruction time per-iteration can be reduced (as compared to ART), at the expense of more iterations being required to find a solution (Dobbins and Godfrey (2003)).

Use of optimization approaches also allows extra sources of information to be incorporated in the form of constraints (Herman (2009)). Examples of such constraints includes

- non-negativity (i.e., absorption coefficient values must be positive)
- *a priori* information

Statistical Methods

Statistical reconstruction methods are a subset of the optimization methods. The differentiating features of statistical approaches include how the matrix A in equation A.3 is defined and how the uncertainties associated with the reconstruction problem are handled. Methods such as ART and SIRT do not consider the uncertainty of the detector measurement or other sources of error, such as scattered radiation. ART and SIRT assume that the measurements (**b** in equation A.3) are due to photons following a straight path and the detector being a perfect photon-counter. The violation of these assumptions results in both methods producing noisy results if algorithm tuning parameters are not carefully controlled.

Sonka and J. M. Fitzpatrick (2000) provide an excellent overview of statistical reconstruction methods and includes the derivations and equations necessary for implementing the various algorithms. Selected approaches will be summarized here, albeit with much of the mathematical detail removed. Unless specifically noted, information in the following sections is taken from Sonka and J. M. Fitzpatrick (2000).

Maximum Likelihood

Many statistically-based reconstruction approaches fall under the broad category of "maximum likelihood". In these approaches a function is defined which places a numerical value on the likelihood that the current reconstruction result is correct given the known measured data. If one thinks of the reconstruction as representing an object of varying attenuation coefficients, the likelihood function answers the question, “What is the probability that this object (our current reconstruction estimate) produced these measurements (the values measured on the detector during the inspection)?”. Logically, when such a probability is maximized the resulting reconstruction is the “best guess” at the true result.

Expectation-Maximization

Expectation-maximization (EM) is a subset of maximum likelihood algorithms. When there is incomplete knowledge of our data (e.g., missing or incomplete data, hidden or unobservable data, etc.) the expected values of the missing data are first calculated in what’s called an “E-step”. Next, using the full dataset, part of which is truly known and part of which is unknown but for which the expecte values are known, the unknown system parameters which maximize the likelihood function are calculated in what’s called an “M-step”. Performing one E-step and one M-step constitutes a single iteration. With a properly-designed algorithm, it can be shown that with each iteration the likelihood associated with our estimation of the unknown model paramters ($\mathbf{x}$ in equation A.3) will either remain steady or rise (i.e., our estimate never gets worse).

- E-step: The expected value of the missing data is calculated. Mathematically, $E[X]$ is calculated, where X is our set of missing data. In the case of tomography X has a Poisson distribution due to the counting of discrete photons (Barrett and Swindell (1981)).

- M-step: With all data now known, either from being truly known or knowing the expected values, the model parameters are maximized for the current values of the missing data.

EM algorithms for both emission and transmission tomography have been developed, but suffers from increased computational time (as compared to other methods) and difficulty in capturing the correct statistics when considering the case of transmission tomography and are better-suited for emission tomography (Sonka and J. M. Fitzpatrick (2000); Lange and Carson (1984)).

APPENDIX B. DETERMINATION OF CUDA RUNTIME CONFIGURATION

Terminology

CUDA programming requires the use and understanding of several terms unique to the GPU programming environment. These terms are defined below.

Compute Capability Single number of the form X.Y which identifies the hardware resources and limitations associated with the device.

CUDA Cores Low-level processor cores which perform the actual calculations of the kernel function. Tens or hundreds (depending on the card) of these CUDA cores can be associated with a single multiprocessor.

CUDA Kernel / Kernel Function which executes on the GPU.

Constant Memory Limited memory resource which is available to all threads. Read-only access allows fast access by threads due to dedicated hardware acceleration. Values stored in this memory must be set from the host.

Device The GPU device. Device code runs on the GPU.

Global Memory Largest block of memory on the device, often multiple gigabytes. This memory is the slowest to access, but is available to all threads.

Host The computer in which the GPU device is installed. Host code runs on the CPU.

Multiprocessor High-level processor which handles the scheduling of thread warps. Often abbreviated as “MP”.

Occupancy Ratio between the actual number of threads running on a multiprocessor and the maximum-possible number of threads running on a multiprocessor.

Registers Memory local to a thread. This includes the value of variables and statically allocated arrays (e.g., `float my_array[6];`).

Shared Memory Limited memory space which is accessible by all threads within a single block. Threads of one block cannot access the shared memory of another block. This memory is local to the MP and is very fast for the threads to access. Shared memory can be statically allocated within the kernel function or dynamically allocated when the kernel is launched.

Thread Block A collection of CUDA threads which have access to a single shared memory region.

Warp A group of threads which are run simultaneously. Currently (May 2013), warps always contain 32 threads.

Warp Divergence Situation when threads in a warp execute different regions of code. This commonly occurs due to if-statements or other instruction-branching points, and it is very detrimental to the overall performance.

GPU Architecture

Note: much of this section is taken from NVIDIA's CUDA-C programming guide (`http://docs.nvidia.com/cuda/cuda-c-programming-guide`). Tthe most- important portions are discussed here for determination of the runtime configuration.

Begin by considering the CUDA threads themselves. These are the workers which run in the CUDA cores and execute the code contained in the kernel. These threads are grouped into "blocks". A single block of threads has access to a limited amount of fast-access memory which is shared amongst all the threads in the block. These blocks are then assembled into a grid. Graphically, this can be seen in figure B.1.

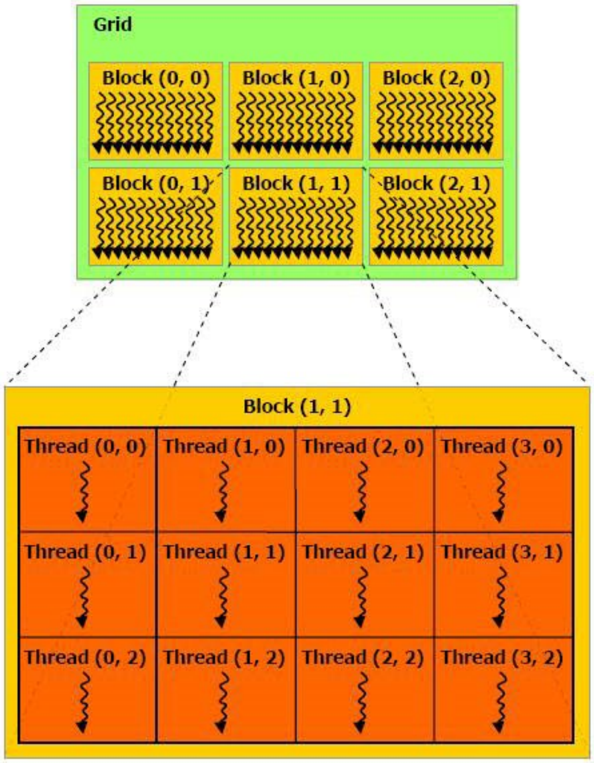


Figure B.1 Grid of thread blocks.

CUDA is based on SIMT (Single-Instruction, Multiple Thread) programming. This means that all threads perfom the same calculations, and that minimizing warp-divergence is necessary to maximize performance.

When writing the CUDA kernels, the data on which a thread performs calculations is identified by the thread's position within the grid. In many cases, a global ID number may be sufficient, while in others the thread's position within the block, and/or the block's position within the grid may be important. The means by which the problem work-units are defined is left to the programmer to decide, and is dependent on the characteristics of the specific algorithm.

The thread blocks can be created as a 1D, 2D, or 3D collection of threads. Depending on the device compute capability, the grid can be either 2D or 3D. Here, dimensionality refers to the number of indices used to identify the thread's position within its block, and the block's position within the grid.

Dimensionality of the blocks and grid has no inherent purpose, other than to facilitate the logical mapping of the algorithm to the GPU architecture. For example, there is no functional difference between the following block configurations:

- 1024x1x1, 1024 threads total
- 512x2x1, 1024 threads total
- 512x1x2, 1024 threads total
- 1x128x8, 1024 threads total
- . . .

Depending on the algorithm, a particular block configuration may make greater logical sense than another. Similarly, the grid configuration is an arbitrary choice by the programmer and can be chosen to best-fit the particular algorithm being implemented.

Limiting Resources

Regardless of the device, the following quantities control the runtime configuration needed by the kernel:

- Registers
- Shared memory
- Allowable dimensions of thread blocks and grid

The available quantities of these resources is controlled by what NVIDIA calls the compute capability of the device. The important factors associated with several compute capabilities are listed in table B.1.

Note that additional memory limitations exist due to the available quantities of global, constant, and texture memories. Addressing these limitations is often as simple as dividing the work into sub-tasks and then looping through those sub-tasks. Such looping won't be discussed in this chapter, and instead the chapter will focus on the more-complicated optimization of the thread-block configuration.

Table B.1 GPU specifications for multiple compute capabilities.

Quantity	**CC 1.3**	**CC 2.0**	**CC 3.5**
32-bit Registers per MP	16384	32768	65536
Register Allocation Size	512	64	256
Max Registers per Thread	128	63	255
Warps per MP	32	48	64
Blocks per MP	8	8	16
Max Threads per MP	1024	1536	2048
Shared Memory per MP	16 kiB	48 kiB	48 kiB
Grid dimensionality	2D	3D	3D
Max Grid Size (any dim.)	65535	65535	$2^{31} - 1$
Architecture Name	Tesla	Fermi	Kepler
Example Device	Tesla C1060	Tesla C2075	Tesla K20c

Optimization Approach

As stated in the NVIDIA CUDA C Programming Guide, there are three primary approaches to use when optimizing a CUDA kernel.

1. Maximize parallel utilization

2. Maximize memory throughput

3. Maximize instruction throughput

In this document the focus will be maximizing the parallel utilization of the kernel. There are two main ways of doing so:

1. Write the algorithm such that the work-units are as small as possible. This will create greater numbers of work units to be split across the CUDA multiprocessors and make the work load as evenly distributed as possible. This is often opposite of designing a parallel algorithm to be run on the CPU.

2. Launch the CUDA kernel with parameters which maximize the rate at which work can be done on the GPU.

Key assumption: increasing occupancy is a good rule-of-thumb for optimizing performance.

Using this key assumption, the optimization criteria is defined as the expected occupancy for a given runtime configuration. With the criteria defined, it is now possible design a process by which the maximum occupancy may be achieved.

Sample-Code

To discuss the optimization approach, consider the sample-code in listing B.1. The code has two parts: querying the CUDA kernel to obtain the resource requirements, and then the actual optimization of the runtime configuration.

```
extern ”C”
void Query_BilateralFilter2Df(int &binaryversion, int &ptxversion,
                              int &maxthreadsperblock, int &numregs,
                              size_t &constsize, size_t &localsize,
                              size_t &sharedsize)
{

    cudaFuncAttributes attrib;

    cudaError_t result = cudaFuncGetAttributes(&attrib,
                           CUDA_FilterBilateral::BilateralFilter2Df);

    binaryversion = attrib.binaryVersion;
    ptxversion = attrib.ptxVersion;
    maxthreadsperblock = attrib.maxThreadsPerBlock;
    numregs = attrib.numRegs;
    constsize = attrib.constSizeBytes;
    localsize = attrib.localSizeBytes;
    sharedsize = attrib.sharedSizeBytes;
}
```

```
extern "C"
void CalcThreadsPerBlock2(size_t maxthreadsperblock, size_t sharedmemperblock,
                          size_t regperthread, devinfo *info,
                          size_t &nthreads, float &occupancy, const char *desc
                          )
{
    /* Algorithm overview:
     *  1.) Get initial value for number of threads by requiring
     *       'maxthreadsperblock' to be a multiple of warp size.
     *  2.) For this initial thread quantity, calculate the occupancy.
     *  3.) Loop through other possible numbers of threads per block, reducing
     *      the number of threads by a warp each loop iteration.
     *      a.) If the iteration is hardware-allowable (i.e., appropriate
     *          register usage, appropriate shared-memory usage, appropriate
     *          number of blocks and warps per multiprocessor), calculate the
     *          occupancy.
     *      b.) If the occupancy is greater than the previous value, save the
     *          current number of threads per block as the new optimum.
     *  4.) If no number of threads per block is found which satisfies the
     *      hardware requirements, return a block size and occupancy of 0 as
     *      an indication that no valid configuration was found.
     */



    /* Set 'nthreads' based on max-allowed threads per block (ignoring shared
     * memory concerns)
     */
    nthreads = maxthreadsperblock;

    /* Require 'nthreads' to be a multiple of 'warpsize' */
    int diff = (int)(nthreads % info->warpsize);
    nthreads -= diff;
    int nwarps =(int)(nthreads/info->warpsize);

    /* Calculate 'occupancy' */
    occupancy = (float)nthreads/(float)info->threadspermp;

    /* Save initial value of 'nthreads'. */
    int nthreadssave = (int)nthreads;

    /* Initialize boolean variables to track if requirements are met. */
    bool valid_regs = true;                /* Checks register usage. */
    bool valid_shared_mem = true;          /* Checks shared-memory usage. */

    /* Loop through the removal of warps from each block and check occupancy
     * on each iteration
     */
    for(int i=0; i<nwarps-1; i++){
        valid_regs = true;

        /* Calculate number of threads per block. */
        int nthreadstmp = (int)nthreads - (int)info->warpsize*i;

        /* Calculate number of blocks possible per multiprocessor for
         * 'nthreadstmp' threads per block
         */
        int nblocks = (int)info->threadspermp/nthreadstmp;

        /* Calculate number of registers required per multiprocessor and check
         * against hardware limit.
         *
         * Two relevant limits:
         *      1.) Number of registers available per multiprocessor.
         *      2.) Number of registers used per-warp are allocated in set
         *          increments.
         */
        int regs_per_warp = nthreadstmp*(int)regperthread;
        if(regs_per_warp % (int)info->regallocsize != 0){
            regs_per_warp += ((int)info->regallocsize -
                              (regs_per_warp % (int)info->regallocsize));
        }

        int warps_per_block = nthreadstmp/(int)info->warpsize;
        int regreqd = warps_per_block*nblocks*regs_per_warp;

        /* Check if registers required per MP is allowed.  If not, reduce
         * number of blocks per MP until limit is satisfied.
         */
        if(regreqd > (int)info->regspermp){
            valid_regs = false;
            int nblockstry = nblocks - 1;
            while(nblockstry > 0 && valid_regs == false){
                regreqd = warps_per_block*nblockstry*regs_per_warp;

                nblockstry--;

                if(regreqd < info->regspermp){
                    valid_regs = true;
                    nblocks = nblockstry;
                }
            } /* while(nblockstry > 0 && valid_regs == false) */
        } /* if(regreqd > info->regspermp) */



        /* Check shared memory limit of multiprocessor.  Again, reduce the
         * number of blocks per MP until limit is satisfied.
         */
        valid_shared_mem = true;
        if(sharedmemperblock*nblocks > info->sharedmem){
            valid_shared_mem = false;
            int nblockstry = nblocks - 1;

            while(nblockstry > 0 && valid_shared_mem == false){
```

```

            nblockstry--;

            if(sharedmemperblock*nblockstry > info->sharedmem){
                valid_shared_mem = true;
                nblocks = nblockstry;
            }
        } /* while(nblockstry > 0 && valid == false) */
    } /* if(sharedmemperblock*nblocks > info->sharedmem) */


    /* If register-usage OR shared-memory limits are violated, set the
     * number of blocks to 0.
     */
    if(!valid_regs || !valid_shared_mem){
        nblocks = 0;
    }

    /* Ensure that number of blocks is below hardware limit */
    if(nblocks > (int)info->maxblockspermp){
        nblocks = (int)info->maxblockspermp;
    }

    /* Check that number of warps is below hardware limit */
    if(warps_per_block*nblocks > (int)info->warpspermp){
        nblocks = (int)info->warpspermp/warps_per_block;
    }

    /* Check occupancy for current values */
    float occupancytmp = (float)nthreadstmp*(float)nblocks/
                                                (float)info->threadspermp;

    /* If the occupancy improves, and all hardware limits are satisfied,
     * update the number of threads per block and the corresponding
     * occupancy.
```

```
166          */
167         if(occupancytmp > occupancy && valid_regs && valid_shared_mem){
168             nthreadssave = nthreadstmp;
169             occupancy = occupancytmp;
170         }
171     } /* Loop over warps-per-block. */
172
173     nthreads = nthreadssave;
174 }
```

Listing B.1 Optimization sample-code

Kernel Resource Query

We see the function to query the resource requirements begining on line 2. The CUDA API contains the actual function query (line 10), and it is contained within a wrapper function to allow access from code not compiled with the CUDA compiler.

When optimizing the runtime configuration, only a portion of the queried information is of-interest:

- Maximum number of threads per block.
- Number of registers used per thread.
- Quantity of statically-allocated shared memory per block.

The maximum number of threads per block provides a starting point for the optimization calculations. Attempting to use more threads than this per block will cause the kernel launch to fail.

The register usage is commonly a limiting factor due to the limited number of registers available per multiprocessor. This has a significant effect on the number of blocks which may reside concurrently on a single multiprocessor, and thus the occupancy which may be achieved.

Shared memory is another scarce quantity on the multiprocessor. Querying the function resource usage returns the *statically-allocated* shared memory. Any shared memory which is dynamically allocated when the kernel is launched is not included in this total, and must be accounted-for manually.

Optimization Algorithm

We can now discuss the optimization algorithm itself, defined in listing B.1, beginning on line 27.

Overall Approach

The optimization approach used is a brute-force method. Several potential configurations are checked, and the one which maximizes occupancy is defined to be the optimum. This is due to the earlier assumption that increased occupancy results in increased performance.

The algorithm itself is rather straightforward.

1. Beginning with the maximum possible threads per block (found via the kernel query function results), determine how many blocks can run concurrently on a single multiprocessor. This is limited by two hardware resources: registers and shared memory.

    - Calculate the number of blocks which will satisfy the register limit.
    - Using the register limit as a starting point, calculate the number of blocks which will satisfy the shared memory limit.

2. If both limitations are satisfied, calculate the occupancy.

3. If the occupancy is greater than the current best-value, save the current number of threads per block as the current optimum.

4. Loop through all possible block sizes and repeat the above process.

Although the algorithm is rather straightforward, there are a couple key points to be expanded upon.

Key Point: Allowable Block Sizes

CUDA kernels are executed in groups of 32 threads called warps. As a result, optimum performance occurs when an integer number of warps are used. This means that when looping through the possible block sizes, one warp of threads must be removed from the count on each iteration.

Execution may be possible with non-integral warp-counts, but performance will suffer. As a result, such cases are not considered.

Key Point: Register Allocation Size

A tricky point when calculating the optimum configuration comes when calculating the number of registers used by the block of threads. If each thread requires X registers, and there are Y threads in the block, one would think that the block would use XY registers from the multiprocessor. This is not necessarily the case.

Registers are allocated from the multiprocessor on a warp-by-warp basis, and the allocation can only occur in specify sizes. This is called the *register allocation size*, and is dealt with on line 94 of the code listing. Note that the register allocation size varies with compute capability.

As a concrete example, consider a kernel which requires 30 registers per thread. This means that each warp will require 30 registers/thread * 32 threads/warp = 960 registers/warp. Consider three different compute capabilities (1.3, 2.0, and 3.5) to see how the changing compute capability affects the register usage. Note that as the compute capability changes one has to consider 3 different changing values:

- register allocation size
- number of available registers on each multiprocessor
- number of warps which may run concurrently

Compute Capability 1.3

For compute capability 1.3 the register allocation size is 512. Thus, if 960 registers are required, 1024 registers will be allocated and 64 registers are allocated but unused.

With 1024 registers allocated per warp, 16 warps may run concurrently on a single multiprocessor due to the total number of available registers (16384 registers / 1024 registers/warp = 16 warps). Since each multiprocessor can have up to 32 warps per multiprocessor, this configuration would only achieve 50% occupancy.

Compute Capability 2.0

Repeat the calculations using different hardware. The register allocation size for compute capability is 64. This means that 960 registers will be allocated and there are no "wasted" registers.

With 960 registers per warp, and 32k registers per MP, each MP can now run 34 warps. Each MP can have up to 48 warps, so 70% occupancy can be achieved simply by using different hardware.

Compute Capability 3.5

For one final example, consider the most-recent (as of this writing) compute capability. Registers are allocated in groups of 256, and 64k registers are available on each MP.

Due to the allocation size, 1024 registers will be allocated and like in the first example there will be 64 "wasted" registers. With 1k registers allocated per warp, there are enough available registers to support 64 warps running concurrently. This corresponds to a perfect occupancy of 100% on this hardware.

Implementation

When integrating CUDA functionality into code there are several goals in mind

- Keep the CUDA code fully separated from the rest of the code. This modularity helps with reusing code in other locations and avoids unnecessary usage of the CUDA compiler.
- Automatically handle the division of the workload into sub-tasks if the user-specified GPU cannot process the entire workload at once.

- Automatically determine the runtime configuration for the kernel which will maximize the kernel's occupancy.

The first two goals are accomplished by writing a wrapper function which surrounds the kernel. The wrapper function is contained within the *.cu source file which also contains the kernel code. It is delcared *extern "C"*, allowing it to be linked against objects compiled by the non-CUDA compiler. Since the CPU code does not directly call launch the kernel, no CUDA functionality is introduced to the CPU code, and thus the CPU does not need to be compiled with the CUDA compiler.

The wrapper also divides the workload into sub-tasks as-necessary in order to allow the GPU to be used to perform the calculations. This aleviates many potential headaches when trying to write porable code that requires minimal modifications (ideally, no modifications) when running the kernel on various GPU devices. The work-load division is primarily driven by the available global memory on the device.

To accomplish the third goal, a device-information structure has been created to describe the GPU device and written functions to query the device properties and perform the optimization described above. Use of a custom device-information structure is necessary to combine properties obtained through several CUDA API functions/structures. By using a custom structure a single object can be used to describe all relevant details of the device, and a consistent structure may be used across all of the custom CUDA functions, both in CPU code and GPU code. The structure itself has no CUDA-specific attributes, allowing it to be used in CPU code without requiring the use of the CUDA compiler. This structure is defined in listing B.2.

```
/** @brief Structure which contains information about a CUDA-capable device. */
struct devinfo {
    /** @brief Device name. */
    char name[256];

    /** @brief Total memory available on device, in Bytes. */
    size_t totalmem;

    /** @brief Free memory available on device, in Bytes. */
    size_t freemem;

    /** @brief Constant memory available on device, in Bytes. */
    size_t constmem;

    /** @brief Shared memory available per block, in Bytes. */
    size_t sharedmem;

    /** @brief Compute-capability of the device. */
    float computecap;

    /** @brief Maximum grid size in each dimension. */
    size_t maxgridsize[3];

    /** @brief Maximum block size in each dimension. */
    size_t maxblocksize[3];

    /** @brief Registers available per block. */
    size_t regperblock;

    /** @brief Threads available per block. */
    size_t threadsperblock;

    /** @brief Threads allowed per multi-processor. */
    size_t threadspermp;

    /** @brief Number of threads in a warp. */
    size_t warpsize;

    /** @brief Maximum number of blocks allowed per multi-processor. */
    size_t maxblockspermp;

    /** @brief Maximum number of warps allowed per multi-processor. */
    size_t warpspermp;

    /** @brief Maximum number of registers allowed per multi-processor. */
    size_t regspermp;

    /** @brief Register allocation size. */
    size_t regallocsize;

    /** @brief Number of multiprocessors. */
    size_t num_mp;

    /** @brief Version of driver API used for device. */
    float api_ver_driver;

    /** @brief Version of runtime API used for device. */
    float api_ver_runtime;
};
```

Listing B.2 Device information structure

Most of the structure member variables in listing B.2 are found using the CUDA API function *cudaGetDeviceProperties()*. This function, however, does not provide information about the device memory (total and free sizes) and also does not provide information about some of the compute-capability-dependent parameters (e.g., registers available per multiprocessor). These missing values are found using other API functions and a function written by me which sets values based on the compute capability.

The end-result of using wrapper functions, custom device information structure, and run-time configuration optimization algorithm is that CUDA code can be run across nearly all (sufficiently old and/or "weak" GPUs may be incapable of running the code) NVIDIA GPUs. The varying GPUs does not require any extra input on the part of the user, and the kernel occupancy should be maximized, regardless of the GPU selection. This is a significant step towards writing effective, portable GPU code that works across the spectrum of GPU devices, for both consumer-grade and high-performance devices. Such behavior is important in enabling effective research tools and paramount when integrating GPU acceleration into the commercial simulation tools.

APPENDIX C. GPU IMPLEMENTATION OF THE BILATERAL FILTER

The Bilateral Filter

Continuous Form

For later reference, here is the expression for the bilateral filter from Tomasi and Manduchi (1998).

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbf{f}(\xi) c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \tag{C.1}$$

$$k(\mathbf{x}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, \mathbf{x}) s(\mathbf{f}(\xi), \mathbf{f}(\mathbf{x})) d\xi \tag{C.2}$$

In these equations **f** represents the value of the image data, c is the *closeness function* which weights values based on spatial proximity, and s is the *similarity function* which weights values based on the similarity of their intensities. The variable ξ is used as a position vector within the filter template, and **x** is the position vector within the image of the pixel being filtered. Note that the “filter template” can also be called the “filter kernel”. The term “template” is used here to try and avoid potential confusion with the CUDA kernel which will be discussed later.

In equation C.1 the filter template is expressed as being the entire image due to the infinite limits of integration. From a practical point of view, it is desireable to limit the filter to only consider a restricted neighborhood of pixels around the pixel which is being filtered. Tomasi and Manduchi proposed the use of exponentials for the closeness and similarity functions. The decay behavior of the exponential allows the filter to discard pixels which are sufficiently far away from the center of the filter template.

Discrete Form

We must also express equations C.1 and C.2 as discrete sums in order to consider their implementation in a computational scheme. Using the weighting functions proposed in Tomasi and Manduchi (1998), the filter equations can be expressed as a summation as follows.

$$\mathbf{h}(\mathbf{x}) = k^{-1}(\mathbf{x}) \sum_{m=0}^{M} \sum_{n=0}^{N} \mathbf{f}(\xi) e^{-\frac{1}{2}\left(\frac{d(\xi,\mathbf{x})}{\sigma_D}\right)^2} e^{-\frac{1}{2}\left(\frac{\delta(\mathbf{f}(\xi),\mathbf{f}(\mathbf{x}))}{\sigma_R}\right)^2} \tag{C.3}$$

$$k(\mathbf{x}) = \sum_{m=0}^{M} \sum_{n=0}^{N} e^{-\frac{1}{2}\left(\frac{d(\xi,\mathbf{x})}{\sigma_D}\right)^2} e^{-\frac{1}{2}\left(\frac{\delta(\mathbf{f}(\xi),\mathbf{f}(\mathbf{x}))}{\sigma_R}\right)^2} \tag{C.4}$$

$$d(\xi, \mathbf{x}) = ||\xi - \mathbf{x}|| \tag{C.5}$$

$$\delta(\phi, \mathbf{f}) = ||\phi - \mathbf{f}|| \tag{C.6}$$

Equations C.3 and C.4 are written for a rectangular $M \times N$ template. Typically, M and N are odd so that the pixel being filtered is centered in the template. Within this paper, the term “central pixel” will be used to refer to the pixel which is being filtered.

Equation C.5 is simply the Euclidean distance between two points, and equation C.6 is the magnitude of intensity difference between two points.

Computational Implementation

Before the discrete equations can be implemented, their parameters must be fully defined. This implementation will be using the exponential weighting functions proposed by Tomasi and Manduchi (1998), but alternate functions may be used.

There are two user-controlled parameters in this expression of the filter: σ_D and σ_R. These control the shape of the exponential weighting functions and thus the inclusivity of the weighting. Smaller values will require pixels to be closer, both in space and intensity, in order to have much contribution to the sum. Larger values will result in a more-inclusive behavior as pixels which are located farther away are allowed to contribute, as well as pixels which have greater intensity differences from that of the central pixel.

Spatial-Weighting Behavior

As defined here, the kernel sizes N and M appear to be additional user-controlled parameters. However, once N and M become sufficiently large, the outer regions of the template will have no appreciable effect due to the closeness function. Thus, the template size is effectively linked to σ_D. By defining a threshold for the closeness function, after which its magnitude is deemed negligible, a relationship between the template size and σ_D can be expressed.

For a threshold value of T, the distance required to reach this threshold can be calculated.

$$d = \sqrt{-2ln(T)\sigma_D^2} \tag{C.7}$$

By using d from equation C.7 as the half-size of the template, M and N can be calculated.

$$M = 2d + 1 \tag{C.8}$$

$$N = 2d + 1 \tag{C.9}$$

Conversely, d can be defined by the user, and then σ_D can be calculated.

$$\sigma_D = \sqrt{\frac{-d}{2ln(T)}} \tag{C.10}$$

As a concrete example, consider a threshold of 1% ($T = 0.01$). Using equations C.8 and C.9, the template will be sized such that the outer-most pixels will only contribute 1% due to the closeness weighting. The corners of the template will contribute less due to their greater distance to the center as-compared to the edge-centers, and values outside the template would contribute even less if they were to be considered. The justification behind setting a threshold such as this is that such low weightings are likely to have a negligible effect on the sum and the difference in result is not worth the added computational effort associated with a larger template.

The effect of template size as a function of σ_D can be seen in figure C.1.

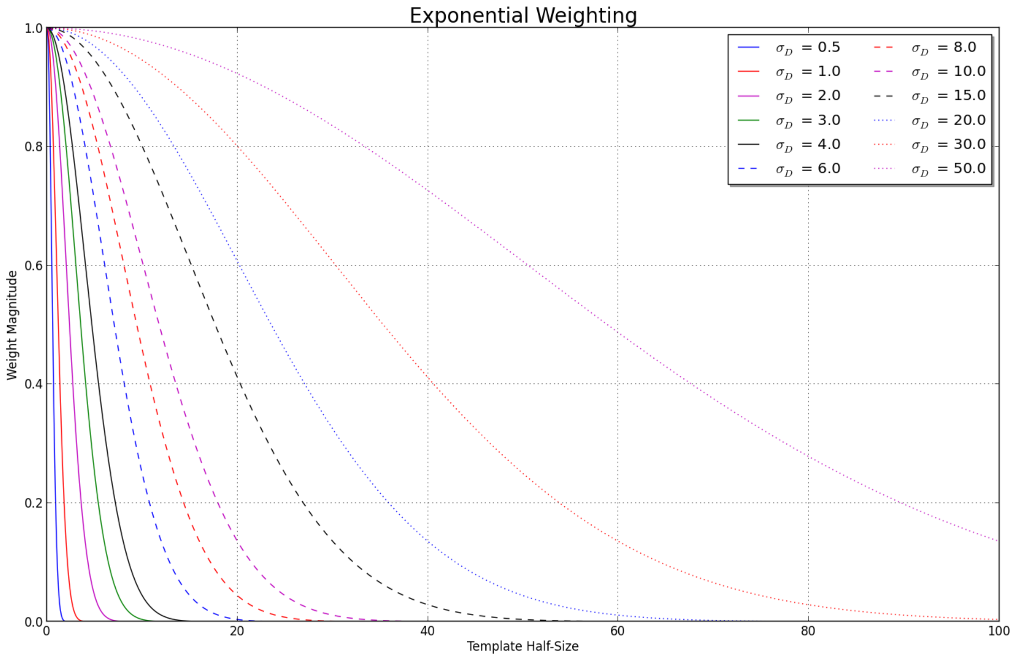


Figure C.1 Template half-size as a function of σ_D

Intensity-Weighting Behavior

We have yet to consider σ_R, the parameter controlling the weighting as a function of intensity similarity. This parameter can be set one of two ways:

1. As an absolute value. This requires the user of the filter to have advance knowledge of the intensity variance in the image and also assumes that the variance is uniform throughout the image.

2. As a scale factor for the actual data variance. This requires less *a priori* knowledge on the part of the user at the cost of additional calculations during the application of the filter.

From a practical perspective, approach 2 is much more robust. The additional calculations required are due to the need to calculate the variance of the intensities within the kernel template. The user-supplied σ_R is then used as a multiplicative scale factor with this actual variance.

The final quantity which must be defined is the reference value used when comparing intensities. This is the quantity ϕ in equation C.6. The simplest specification is

$$\phi = |\mathbf{f}(\mathbf{x})| \tag{C.11}$$

However, if the bilateral filter is being used to remove noise in the form of outlier intensities which are significantly different from their neighbors (e.g., dead pixels in an x-ray detector), such a specification of ϕ may be problematic. The problem arises from the fact that all pixels in the template will have significantly different intensities from the reference value, which will cause the weights associated with each pixel to be very small. Thus, the weighted sum will produce a result effectively identical to the pixel's initial value, meaning that no filtering was actually performed.

To mitigate this issue, the reference value, ϕ, is calculated as the average value of the pixels in the kernel, excluding a region near the center. Excluding a 3 x 3 region, for example, prevents a small cluster of outlier pixels from detrimentally affecting the calculation. If such a region is excluded when defining the reference value, it is also excluded when calculating the variance of the kernel values.

GPU Implementation

Preliminary Considerations

The GPU platform uses hundreds, possibly thousands (depending on the device), of lightweight threads running in parallel. Efficient execution of the CUDA kernel requires that these threads run in a coordinated fashion. This coordinate has two aspects to it:

1. Instructions are identical between threads, allowing thread groups, called warps, to execute in lock-step. Differing instructions between threads results in what is called warp-divergence and can lead to significant performance losses.

2. Threads access memory without conflicting with one another. This means the algorithm must avoid having multiple threads attempt to access the same memory location.

Additionally, the algorithm must minimize the references to global device memory due to the significant latency associated with such operations.

When designing the algorithm to run on the GPU the programmer must consider the execution structure of the device. Individual threads are grouped into blocks, and the blocks are in turn grouped into a grid. This is shown graphically in figure B.1.

My Chosen Implementation

Preliminary Comments

With the device architecture in-mind, the programmer can now consider the implementation of the bilateral filtering algorithm. It should be noted that alternative implementations have not been explored, and faster implementations may exist. This algorithm has been implemented with the following objectives:

1. Implement the filter while taking advantage of easily-found optimizations. In-depth optimization is left for future work.

2. Algorithm clarity. The code should be written in a manner which allows someone else to understand the algorithm and its implementation.

3. Extension to 3D. Three-dimensional filtering is an eventual goal, and an algorithm which is easily extended from 2D to 3D will make that transition easier as well as make code-maintenance and debugging easier.

Simple Implementation: One Pixel per Block

Now consider the actual implementation of the algorithm. We'll begin with the simple case of having each thread block process a single pixel of the image. A more-complex case will be discussed in section C.

In each case the descriptions here will focus on the bilateral filter itself. The additional calculations to determine the reference value, ϕ, and the variance of the kernel values are conceptually straightforward. Further, these calculations are implmented in a similar means as the filter itself, allowing a single explanation to be used to explain them all.

Block Structure We define each thread block to be a 2D array of CUDA threads. The dimensions of the block are chosen to be $(M+1) \times (N+1)$, where M and N are the template dimensions as defined above. Since M and N are odd, an extra thread is added in each dimension so that the block dimensions are even. Even dimensions are needed when performing the parallel summation (discussed on page 135, under the heading **Parallel Summation**).

Using a 2D block of threads for the kernel exchanges a double-loop through the kernel (ref. eq. C.3) for MN threads running in parallel. The massive number of CUDA threads running on the device makes such an exchange feasible. Modern CPUs can handle O(20) concurrent calculational threads, and are thus better suited for the double-loop approach.

Shared Memory Usage Evaluating equation C.3 requires summing two quantities:

1. The weight factor applied to each element within the template.
2. The product of the weight factor and the elemental value for each element within the template.

The filter result is then simply the result of dividing the sum of the weight products by the sum of the weights.

When performing the calculations within a double-loop, such as on a CPU, the algorithm simply needs to declare a variable for each of these quantities and update a running sum on each loop iteration. Since the algorithm has replaced the double loop with a 2D block of threads an alternative approach is necessary. Each thread only performs calculations for a single element within the filter template and has no knowledge of the data and weight values calculated in a different thread. An efficient means of inter-thread communication is needed.

This is accomplished by using the shared memory on the GPU. Shared memory is a limited quantity of fast-access memory which is available to all threads in a block. Shared memory cannot be shared between blocks. The available shared memory per multiprocessor varies by device, but currently (May 2013) ranges between 16 kiB and 48 kiB. If multiple blocks are running concurrently on the multiprocessor they must pull their shared memory from the same bank of available memory. For example, if 2 blocks are running concurrently on a device with 16 kiB of shared memory per multiprocessor, each block must use 8 kiB or less.

Sample code is shown in listing C.1, and at the end of the code it is seen the weight value and weight-data product being placed into a shared memory array called `kernel_data`. The values in shared memory can then be summed (discussed later).

The shared memory array is sized to have two elements per thread in the block. One element is for the weight factors and the other is for the product of the weight factors and the data. The elements are grouped with the products in the first half of the array and the weights in the second half of the array. This structure is used due to the way in which the array must be allocated. Since the array size is not static (i.e., it is defined at runtime rather than compile-time), the shared memory must be declared as a single 1D array. The array size, in Bytes, is specified as one of the kernel-launch parameters.

Since the shared memory size is based on the number of threads in the block, once the block size is maximized, the shared memory usage will cease to grow. This independence from template size is beneficial for large templates (handling of large templates is addressed later).

Filter Calculations Sample code for the filter calculations can be seen in listing C.1. This is a snippet from a much larger function. Values used in this function, but defined in earlier code (and thus not shown here), are identified in the comments at the beginning of the listing.

```
/* 'kernel_data' is a shared-memory array initialized earlier in the code
 *
 * 'dist_offset' is an offset that is used to partition the shared memory
 * array into 2 sections
 *
 * (x,y) indices of the central pixel have been calculated earlier in the
 * kernel code
 *
 * 'threadIdx.x' and 'threadIdx.y' are automatically defined by the CUDA API
 * and represent the thread's coordinates within the thread block
 *
 * idx1/2 are the column/row indices of the central pixel of the template
 */

/* Calculate spatial distance between this thread's pixel and the
 * central pixel. */
float dx = (float)(threadIdx.x + idx1) - (float)(idx1 + kernel_dims[0]/2);
float dy = (float)(threadIdx.y + idx2) - (float)(idx2 + kernel_dims[1]/2);

/* Calculate global-memory index containing the data for this thread's
 * template element. */
cuda_uint global_idx = CUDA_Thread::IdxDoubleToSingle(data_dims[0],
                                        threadIdx.x+idx1, threadIdx.y+idx2);

/* Get the value from global memory corresponding to this thread's element. */
float local_value = data_in[global_idx];

/* Calculate the Euclidian distance between this thread's element and the
 * central pixel.  Distance is left squared due to how it is used in the
 * exponential. */
float distsq = dx*dx + dy*dy;

/* Calculate the difference between this thread's element value and
 * the template reference value. */
float value_diff = local_value - center_value;

/* Calculate the weight factor for this element based on its proximity to
 * the central element and the similarity of the intensities. */
float weight = expf(-0.5f*(value_diff*value_diff/var_range +
                                                        distsq/var_domain));

/* If this element is within the kernel, add the weighted-value and the weight
 * itself to the block's shared memory. */
if(threadIdx.x < kernel_dims[0] && threadIdx.y < kernel_dims[1]){
    kernel_data[th_id_block]               = weight*local_value;
    kernel_data[th_id_block + dist_offset] = weight;
```

```
float sum = 0.0f;

for(int i=0; i<array_size; i++){
    sum += kernel_data[i];
}
```

Listing C.2 Naive summation in shared memory.

```
}
```

Listing C.1 Bilateral filter CUDA code

Parallel Summation Placing the weight-data product and weight values into shared memory requires an additional step of summing those values. In order to be performed efficiently, a means of parallel-reduction must be performed. In parallel computing “reduction” is when a quantity distributed across multiple compute-elements (and can thus be considered to exist multiple times) must be “reduced” to a single quantity which may or may not be known to all compute-elements. The term “compute-element” refers to the discrete computational resources which are working together in parallel. In CUDA programming each thread is a “compute-element”, while on a Beowulf cluster each node is a “compute-element”.

A naive means of performing this summation would be to simply use the code in listing C.2.

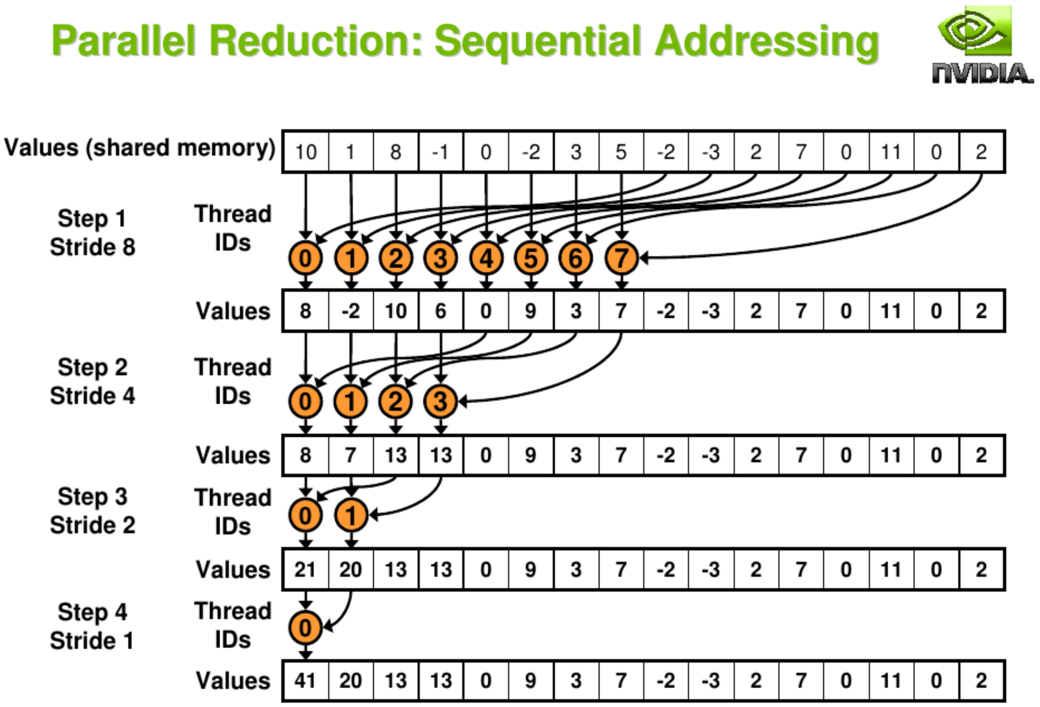
There is a serious problem with this approach. All threads in the block will execute the summation, causing what are known as “bank conflicts”. Bank conflicts are when multiple CUDA threads attempt to access the same memory address at the same time. Only one thread may access an array element at a time, meaning that the code in listing C.2 becomes a bottleneck. Furthermore, the sum is performed in each thread, meaning that the block will perform the sum $(M + 1) \times (N + 1)$ times when only a single summation is required.

A simple solution would be to only let a single thread in the block perform the summation. This is not optimial, either. Summing the array with a simple loop as in listing C.2 requires the thread to perform `array_size` operations. By using multiple threads within the block the summation can be performed in a fraction of the time.

```
for(int s=num_threads_in_block/2; s>0; s>>=1){
    if(thread_id < s){
        kernel_data[thread_id] += kernel_data[thread_id + s];
    }
    __syncthreads();
}
```

Listing C.3 Sequential addressing summation.

The key to efficiently summing these values lies in carefully controlling how the threads access the array so as to avoid bank conflicts. Figure C.2 is taken from a publicly-available NVIDIA presentation by Mark Harris Harris (), and it shows a technique called "sequential addressing". As seen in the figure, summing a 16-element array can be performed in 4 iterations. A single thread executing the code in listing C.2 would require 16 iterations to sum the same array, demonstrating a factor of 4 difference between the two approaches.



Parallel Reduction: Sequential Addressing

	Values (shared memory)	10	1	8	-1	0	-2	3	5	-2	-3	2	7	0	11	0	2
Step 1 Stride 8	Thread IDs	0	1	2	3	4	5	6	7								
	Values	8	-2	10	6	0	9	3	7	-2	-3	2	7	0	11	0	2
Step 2 Stride 4	Thread IDs	0	1	2	3												
	Values	8	7	13	13	0	9	3	7	-2	-3	2	7	0	11	0	2
Step 3 Stride 2	Thread IDs	0	1														
	Values	21	20	13	13	0	9	3	7	-2	-3	2	7	0	11	0	2
Step 4 Stride 1	Thread IDs	0															
	Values	41	20	13	13	0	9	3	7	-2	-3	2	7	0	11	0	2

Figure C.2 Parallel summation using sequential addressing Harris ().

Adapting this approach to the CUDA kernel, code that looks like what is shown in listing C.3 appears. The __syncthreads() command is needed to avoid a race condition. The summation operations for one step must be completed before the next step may be started.

KEY CONFIGURATION NOTE: Proper execution of the parallel summation requires that the number of threads in the XY slice of the thread block be a factor/multiple of the warp size. Through trial and error it was found that block sizes of 4 x 4, 8 x 8, and 16 x 16 work, but other sizes do not. The best performance has been found with the 4 x 4 block dimensions.

Handling Large Templates The above sections describe the key elements of the filter. But the programmer must consider what happens when the filter template grows larger than the allowable thread block dimensions. To motivate this concern, consider that as of May 2013 a single thread block cannot contain more than 1024 threads. This limits a square template to be 31 x 31. When attempting to filter long length-scale trends, 30 pixels is not a particularly long distance.

Additionally, the goal is to eventually want to apply this algorithm to 3 dimensions. A 3D template which is 9 x 9 x 9 will be the largest which can fit within the 1024-thread limit. This template is rather small in all 3 dimensions, requiring the programmer to consider how to handle larger cases.

Recall that shared memory can only be accessed by threads of a single thread block. Even if multiple blocks are running concurrently on a multiprocessor, each block will have its own region of shared memory. This means that the algorithm needs to either "loop" the thread block across the template, or redesign the summation approach. Looping requires minimal modification to the code, and allows a direct extension to 3D.

To loop the block across the template, place the code from listing C.1 within a double-loop. Figure C.3 shows the positions of a 5 x 5 thread block when processing a 7 x 7 template. Two loop iterations are required in each direction, with each iteration shaded a different color.

When implementing this looping approach, the following modifications are required

- When placing values into the shared memory array, replace the assignment operator (=) with addition plus assignment (+=). The shared memory array is still sized to have two elements per thread (1 for the weights and 1 for the weight-data product). Since the shared memory is eventually summed, this summation does not affect the result.

- When calculating the (x,y) coordinates of the thread element (in listing C.1), the loop iteration must be considered. For example, instances of `threadIdx.x` in the sample code become `threadIdx.x + blockDim.x*j`, where `j` is the loop iteration in the x-direction.

- The loops must be performed in three locations:

    1. When calculating the reference value, ϕ, if not using the template's central pixel as the reference.
    2. When calculating the variance of the kernel values.
    3. When calculating the weight factors and weight-data products.

One sacrifice made with this looping approach is that the data value local to each thread cannot be saved between calculation steps. Since each thread may process more than one template element, the value must be pulled from global memory in each step. This results in a maximum of 3 references to global memory per-thread instead of 1. However, some of the additional latency is hidden by the calculations which must be performed in between global memory accesses. No effort has been made yet to quantify the actual time lost by the increased memory accesses.

Runtime Configuration Since each block processes a single pixel of the image, the CUDA kernel must be launched with a grid containing one block per pixel. For the simple case of one pixel processed per block, there is no runtime configuration optimization which may be performed. For small kernels the GPU hardware will likely be severely under-utilized. Such under-utilization is the motivation for the advanced implementation described in section C.

Advanced Implementation: Multiple Pixels per Block

As noted in the previous discussion, the basic implementation will severely under-utilize the GPU resources. Experimental testing has shown this to still be faster than an equivalent CPU algorithm, but with a few modifications to the algorithm the number of threads running in parallel on the device can be dramatically increased.

Block Structure As with the simple implementation, consider a 2D thread block that maps to the filter template. Call this 2D arrangement of threads a “slice” of the thread block. Now, consider the case where there is a 5 x 5 filter template. This results in the slice being 6 x 6, which uses 36 threads. If the block can have 1024 threads, this means there can be 1024/36 = 28 slices of threads in the block.

Thread blocks are allowed to be 3D, so now have the thread block go from 6 x 6 x 1 to 6 x 6 x 28. Each of these 28 slices will process a different pixel in the image. The key step in implementing this is in specifying *which* pixel each slice will process.

One may think that having each slice in the block process the next pixel in the image would be a reasonable approach. The filter templates would have significant overlap which could allow for optimizing global memory access. However, minimizing the global memory accesses becomes a complex bookkeeping task which distracts from the objective of implementing a useful tool and moving-on to use the tool. Additionally, the wrapping from one row to the next would become complex to implement.

The implemented method eliminates the bookkeeping by placing the filter templates for slices adjacent to each other. This is shown in figure C.4. There are no overlapped values to be shared among slices, and wrapping from one image row to the next does not cause any problems.

Figure C.4 shows 7 slices, each 5 x 5, laid out side-by-side. Because the template is 5 pixels wide, there are 4 pixels on the image row between the centers of adjacent slices. These in-between pixels must also be filtered, and assigning thread blocks to them is not too difficult.

We begin by creating a “group” of blocks. The number of blocks in a group is equal to the horizontal size of the filter template. We can see this in figure C.4. If the shaded slices are part of block 0, one can see how block 1 would be centered on the pixels immediately to the right of the central pixels of block 0. Similarly, block 2 would be centered immediately to the right of the block 1 centers. Once block 4 is considered, all of the in-between pixels have been filtered. Block 5, which is the first block in the next group, will appear to continue the slice pattern of block 0 (i.e., the first slice of block 5 will be adjacent to the last slice of block 0).

```
for(int s=num_threads_in_slice/2; s>0; s>>=1){
  if(thread_id_in_slice < s){
      kernel_data[thread_id_in_slice] += kernel_data[thread_id_in_slice + s];
  }
  __syncthreads();
}
```

Listing C.4 Sequential addressing summation for multi-slice implementation.

Parallel Summation Shared memory usage in this multi-slice implementation is very similar to the single-slice implementation. A key difference, however, lies in the parallel summation code. Here, the summation must be performed for each slice as shown in listing C.4. The key change is that the loop is now over the number of threads per-slice rather than per-block, and the thread ID check is based on the ID within the slice rather than within the block.

KEY CONFIGURATION NOTE: Proper execution of the parallel summation requires that the number of threads in the XY slice of the thread block be a factor/multiple of the warp size. Through trial and error it was found that block sizes of 4 x 4, 8 x 8, and 16 x 16 work, but other sizes do not. The best performance has been found with the 4 x 4 block dimensions.

Runtime Configuration With this advanced implementation it may be possible to try and optimize the runtime configuration. An introduction to a basic runtime optimization technique is provided in B. As of this writing, runtime configuration optimization has not been explored with this CUDA kernel.

This multi-slice implementation also uses the block-looping described above. This provides an interesting opportunity to consider the trade-off between smaller slices, with a corresponding increase in the number of slices, and fewer, larger slices. This could potentially be a complicated optimization problem due to its affect on the global memory access patterns. This is also directly related to optimization efforts to maximize occupancy, which will further complicate the process. As a result, proper runtime configuration optimization will be left for future work.

Additional Comments

Anticipated Benefits of My Implementation

It is believed that the implementation described here has many benefits.

- Function templates of any size can be handled. There are no hardware-induced limitations, regardless of the compute capability of the device.
- The extension to 3 dimensions is straightforward. The block-looping process used for large 2D templates will become essential for all but the smallest 3D templates. Implementing the loop over the 3rd dimension should not require significant effort.
- The algorithm can be implemented in a straightforward manner, making the code easily maintained and readable. The multi-slice extension of the algorithm adds appreciable complexity, but the additional bookkeeping is manageable.
- The single-slice implementation has shown 30x speed improvement over the CPU implementation of the algorithm. The speedup factor grows after block-looping is required, reaching over 100x for a 101 x 101 kernel. Initial tests of the multi-slice implementation suggest improvements of 2-4x over the single-slice implementation.
- Block-looping imposes an upper-limit to the time required to perform the parallel-reduction summation. This causes the marginal cost for larger kernels to decrease, which is important for trend-removal in large datasets.

Possible Future Work

Several aspects of this filter and its implementation may be considered for additional work and/or refinement.

- Consider alternative weight functions. The GPU architecture allows increased computational complexity of the weight functions.
- General optimization. Easily-seen optimization has been captured here, but more-robust optimization has been left for later efforts.. Detailed optimization work has not been

done since the intent was to get a working filter which enables new data analysis, not to get a fully-optimized algorithm.

- Study the time lost due to the additional global memory accesses caused by the block-looping setup.

- Consider alternate means of maximizing occupancy for small filter templates.

- Consider assigning blocks to non-adjacent regions of the image. Perhaps this could reduce global memory conflicts.

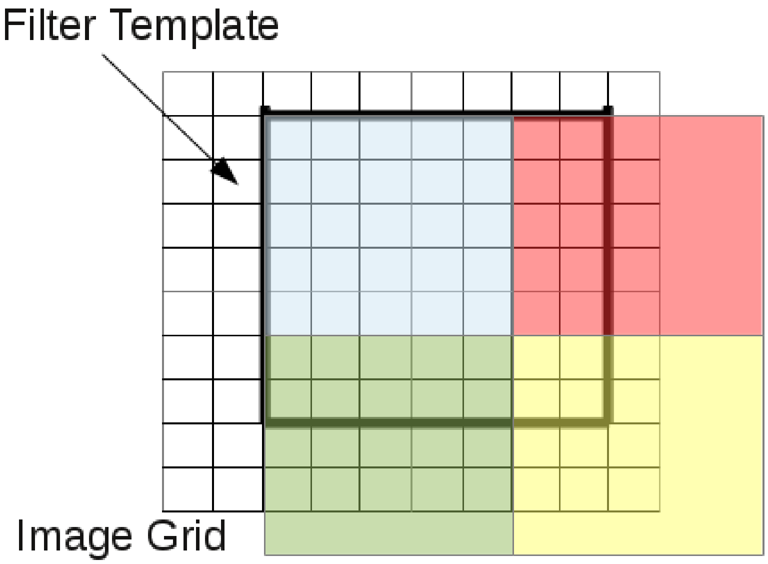


Figure C.3 Looping thread block across template.

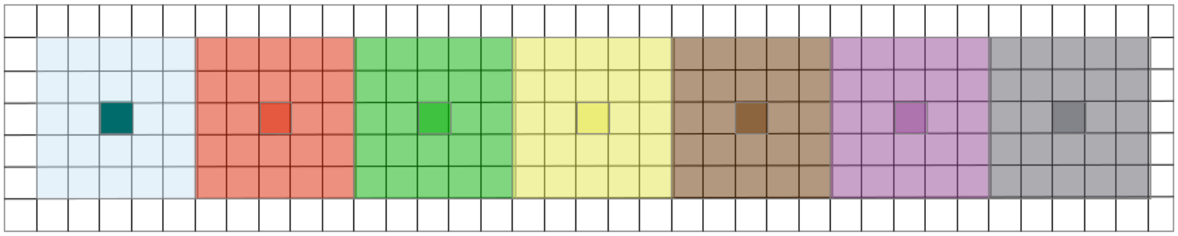


Figure C.4 Adjacent block slices.

BIBLIOGRAPHY

Barrett, H. and Swindell, W. (1981). Radiological Imaging: The Theory of Image Formation, Detection, and Processing. Academic Press, New York, NY.

Bayraktar, E., Antolonovich, S., and Bathias, C. (2006). Multiscale study of fatigue behaviour of composite materials by χ-rays computed tomography. International Journal of Fatigue, 28(10):1322–1333.

Bracewell, R. and Riddle, A. (1967). Inversion of fan-beam scans in radio astronomy. The Astrophysical Journal, 150:427–434.

Bronnikov, A. V. (2000). Cone-beam reconstruction by backprojection and filtering. Journal of the Optical Society of America. A, Optics, image science, and vision, 17(11):1993–2000.

Buffiere, J., Maire, E., and Adrien, J. (2010). In situ experiments with x ray tomography: an attractive tool for experimental mechanics. Experimental mechanics, 50:289–305.

Carlson, W. D. and Denison, C. (1992). Mechanisms of Porphyroblast Crystallization: Results from High-Resolution Computed X-ray Tomography. Science (New York, N.Y.), 257(5074):1236–9.

Cloetens, P., Barrett, R., Baruchel, J., Guigay, J.-P., and Schlenker, M. (1996). Phase objects in synchrotron radiation hard x-ray imaging. Journal of Physics D: Applied Physics, 29(1):133–146.

Cloetens, P., Pateyron-Salome, M., Buffiere, J., Peix, G., Baruchel, J., Peyrin, F., and Schlenker, M. (1997). Observation of microstructure and damage in materials by phase sensitive radiography and tomography. Journal of Applied Physics, 81(9):5878–5886.

Cormack, A. M. (1963). Representation of a Function by Its Line Integrals, with Some Radiological Applications. Journal of Applied Physics, 34(9):2722.

Cormack, A. M. (1964). Representation of a Function by Its Line Integrals, with Some Radiological Applications. II. Journal of Applied Physics, 35(10):2908.

des Plantes, Z. (1932). Eine neue Methode zur Differenzierung in der Röntgenographie. Acta Radiologica, 13:182.

Djukic, L. P., Herszberg, I., Walsh, W. R., Schoeppner, G. a., and Gangadhara Prusty, B. (2009a). Contrast enhancement in visualisation of woven composite architecture using a MicroCT Scanner. Part 2: Tow and preform coatings. Composites Part A: Applied Science and Manufacturing, 40(12):1870–1879.

Djukic, L. P., Herszberg, I., Walsh, W. R., Schoeppner, G. a., Gangadhara Prusty, B., and Kelly, D. W. (2009b). Contrast enhancement in visualisation of woven composite tow architecture using a MicroCT Scanner. Part 1: Fabric coating and resin additives. Composites Part A: Applied Science and Manufacturing, 40(5):553–565.

Dobbins, J. T. and Godfrey, D. J. (2003). Digital x-ray tomosynthesis: current state of the art and clinical potential. Physics in Medicine and Biology, 48(19):R65–R106.

Edholm, P. R. and Herman, G. T. (1987). Linograms in image reconstruction from projections. IEEE transactions on medical imaging, 6(4):301–7.

Fan, P. (2001). High-Resolution CT Data Acquisition Software and 3D Visualization Tool. Master of science, Iowa State University.

Feldkamp, L., Davis, L., and Kress, J. (1984). Practical cone-beam algorithm. JOSA A, 1(6):612–619.

Gao, H., Chen, Z., Xing, Y., and Cheng, J. (2006). An Extrapolation Method for Image Reconstruction from a Straight-line Trajectory. 2006 IEEE Nuclear Science Symposium Conference Record, pages 2304–2308.

Gao, H., Zhang, L., Xing, Y., Chen, Z., and Cheng, J. (2008). An Improved Form of Linogram Algorithm for Image Reconstruction. IEEE Transactions on Nuclear Science, 55(1):552–559.

Garrison, J., Grant, D., Guier, W., and Johns, R. (1969). Three dimensional roentgenography. American Journal of Roentgenology, 105(4):903–908.

Grandin, R. and Gray, J. (2014). Implementation of Automated 3D Defect Detection for Low Signal-to Noise Features in NDE Data. Review of Progress in Quantitative NDE, 1581(1581):1840–1847.

Gray, J., Zhang, J., and Gray, I. (2004). Application of NDE Simulations to Estimate Probability of Detection.

Guvenilir, A., Breunig, T., Kinney, J., and Stock, S. (1997). Direct observation of crack opening as a function of applied load in the interior of a notched tensile sample of Al-Li 2090. Acta materialia, 45(5):1977–1987.

Guvenilir, A., Breunig, T., Kinney, J., and Stock, S. (1999). New direct observations of crack closure processes in AlLi 2090 T8E41. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 357(1761):2755–2775.

Harris, M. Optimizing Parallel Reduction in CUDA. Technical report.

Herman, G. T. (2009). Fundamentals of Computerized Tomography. Springer New York, New, 2nd edition.

Herman, G. T. and Lung, H. P. (1980). Reconstruction from divergent beams: a comparison of algorithms with and without rebinning. Computers in biology and medicine, 10(3):131–9.

Hounsfield, G. N. (1973a). Computerised transverse axial scanning (tomography): Part 1. Description of system. British Journal of Radiology, 46:1016–1022.

Hounsfield, G. N. (1973b). Method and apparatus for measuring x- or gamma-radiation absorption or transmission at plural angles and analyzing the data.

Johnson, H. J., Mccormick, M., Ibanez, L., and Consortium, I. S. (2013). The ITK Software Guide Third Edition - Updated for ITK version 4.5. ITK Project.

Kaczmarz, S. (1937). Angenäherte Auflösung von Systemen linearer Gleichungen. Bulletin de lAcademie Polonaise des Sciences et des Lettres, A35:355–357.

Kini, V. R. (1994). Tomographic inspection system using X-rays. Master of science, Iowa State University.

Kinney, J., Breuning, T., Starr, T., Haupt, D., Nichols, M., Stock, S., Butts, M., and Saroyan, R. (1993). X-ray Tomographic Study of Chemical Vapor Infiltration Processing of Ceramic Composites. Science, 260(5109):789–792.

Koks, D. (2006). Explorations in Mathematical Physics. Springer Science+Business Media,LLC.

Korenblyum, B., Tetel'Baum, S., and Tyutin, A. (1958). About one scheme of tomography. Izvestiya Vysshikh Uchebnykh Zavedenii, Radiofizika, 1:151–157.

Lange, K. and Carson, R. (1984). EM reconstruction algorithms for emission and transmission tomography.

Limodin, N., Salvo, L., Boller, E., Suéry, M., Felberbaum, M., Gailliègue, S., and Madi, K. (2009). In situ and real-time 3-D microtomography investigation of dendritic solidification in an Al10wt.% Cu alloy. Acta Materialia, 57(7):2300–2310.

Limodin, N., Salvo, L., Suéry, M., and DiMichiel, M. (2007). In situ investigation by X-ray tomography of the overall and local microstructural changes occurring during partial remelting of an Al15.8wt.% Cu alloy. Acta Materialia, 55(9):3177–3191.

Liotier, P.-J., Alain, V., and Christine, D. (2010). Characterization of 3D morphology and microcracks in composites reinforced by multi-axial multi-ply stitched preforms. Composites Part A: Applied Science and Manufacturing, 41(5):653–662.

Maire, E., Bouaziz, O., Di Michiel, M., and Verdu, C. (2008). Initiation and growth of damage in a dual-phase steel observed by X-ray microtomography. Acta Materialia, 56(18):4954–4964.

Maire, E., Fazekasb, A., Salvob, L., Dendievelb, R., Youssefa, S., Cloetensc, P., and Letang, J. M. (2003). X-ray tomography applied to the characterization of cellular materials. Related finite element modeling problems. Composites Science and Technology, 63(16):2431–2443.

Maire, E., Morgeneyer, T., Landron, C., Adrien, J., and Helfen, L. (2012). Bulk evaluation of ductile damage development using high resolution tomography and laminography. Comptes Rendus Physique, 13(3):328–336.

Maire, E., Owen, A., Buffiere, J., and Withers, P. J. (2001). A synchrotron x-ray study of a Ti/SiCf composite during in situ straining. Acta Materialia, 49:153–163.

Moffat, A. J., Wright, P., Helfen, L., Baumbach, T., Johnson, G., Spearing, S. M., and Sinclair, I. (2010). In situ synchrotron computed laminography of damage in carbon fibreepoxy [90/0]s laminates. Scripta Materialia, 62(2):97–100.

Natterer, F. and Ritman, E. L. (2002). Past and future directions in x-ray computed tomography (CT). International Journal of Imaging Systems and Technology, 12(4):175–187.

Nobelprize.org (2014a). All Nobel Prizes in Physics.

Nobelprize.org (2014b). All Nobel Prizes in Physiology or Medicine.

Oldendorf, W. H. (1961). Isolated Flying Spot Detection of Radiodensity Dis-Continuities-Displaying the Internal Structural Pattern of a Complex Object. IRE Transactions on Bio-Medical Electronics, 8(1):68–72.

Paris, P. C., Gomez, M. P., and Anderson, W. E. (1961). A rational analytic theory of fatigue. The Trend in Engineering, 13:9–14.

Radon, J. (1917). Uber die Bestimmung von Funktionen durch ihre Integralwerte Langs Gewisser Mannigfaltigkeiten (English translation: On the determination of functions from their integrals along certain manifolds). Ber. Saechsische Akad. Wiss, 29:262.

Ramachandran, G. N. and Lakshminarayanan, a. V. (1971). Three-dimensional reconstruction from radiographs and electron micrographs: application of convolutions instead of Fourier transforms. Proceedings of the National Academy of Sciences of the United States of America, 68(9):2236–40.

Röntgen, W. (1896). On a new kind of rays (Translated by Arthur Stanton from the Sitzungsberichte der Würsburger Physik-medic. Gesellschaft, 1895). Nature, 53(1369):274–277.

Salvo, L., Cloetens, P., Maire, E., Zabler, S., Blandin, J., Buffière, J., Ludwig, W., Boller, E., Bellet, D., and Josserond, C. (2003). X-ray micro-tomography an attractive characterisation technique in materials science. Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms, 200:273–286.

Salvo, L., Suéry, M., Marmottant, A., Limodin, N., and Bernard, D. (2010). 3D imaging in material science: Application of X-ray tomography. Comptes Rendus Physique, 11(9-10):641–649.

Sato, T., Ikeda, O., Yamakoshi, Y., and Tsubouchi, M. (1981). X-ray tomography for microstructural objects. Applied optics, 20(22):3880–3.

Schell, J., Renggli, M., van Lenthe, G., Müller, R., and Ermanni, P. (2006). Micro-computed tomography determination of glass fibre reinforced polymer meso-structure. Composites Science and Technology, 66(13):2016–2022.

Schilling, P. J., Karedla, B. R., Tatiparthi, A. K., Verges, M. a., and Herrington, P. D. (2005). X-ray computed microtomography of internal damage in fiber reinforced polymer matrix composites. Composites Science and Technology, 65(14):2071–2078.

Schwarzband, G. and Kiryati, N. (2005). The point spread function of spiral CT. Physics in medicine and biology, 50(22):5307–22.

Sheikh, N. (2006). Medium resolution Computed Tomography through phosphor screen detector and 3D image analysis. M.s., Iowa State University.

Shepp, L. A. and Logan, B. F. (1974). Reconstructing Interior Head Tissue from X-Ray Transmissions. IEEE Transactions on Nuclear Science, 21(1):228–236.

Snigirev, A., Snigireva, I., Kohn, V., Kuznetsov, S., and Schelokov, I. (1995). On the possibilities of x-ray phase contrast microimaging by coherent high-energy synchrotron radiation. Review of Scientific Instruments, 66(12):5486.

Sonka, M. and J. M. Fitzpatrick (2000). Handbook of Medical Imaging Volume 2: Medical image processing and analysis, volume 2. SPIE.

Spanne, P., Raven, C., Snigireva, I., and Snigirev, a. (1999). In-line holography and phase-contrast microtomography with high energy x-rays. Physics in medicine and biology, 44(3):741–9.

Subramanian, A. (2013). Damage detection and characterization of fiber- reinforced composites using ultrasonics. Master of science, Iowa State University.

Tam, K. C., Lauritsch, G., and Sourbelle, K. (2002). Filtering point spread function in backprojection cone-beam CT and its applications in long object imaging. Physics in Medicine and Biology, 47(15):2685–2703.

Tan, K., Watanabe, N., and Iwahori, Y. (2011). X-ray radiography and micro-computed tomography examination of damage characteristics in stitched composites subjected to impact loading. Composites Part B: Engineering, 42(4):874–884.

Terzi, S., Salvo, L., Suéry, M., Limodin, N., Adrien, J., Maire, E., Pannier, Y., Bornert, M., Bernard, D., and Felberbaum, M. (2009). In situ X-ray tomography observation of inhomogeneous deformation in semi-solid aluminium alloys. *Scripta Materialia*, 61(5):449–452.

Toda, H., Maire, E., Yamauchi, S., Tsuruta, H., Hiramatsu, T., and Kobayashi, M. (2011). In situ observation of ductile fracture using X-ray tomography technique. *Acta Materialia*, 59(5):1995–2008.

Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 839–846.

Wang, G., Lin, T. H., Cheng, P. C., and Shinozaki, D. M. (1992). Point spread function of the general cone-beam x-ray reconstruction formula. *Scanning*, 14(4):187–193.

Weck, A., Wilkinson, D., Maire, E., and Toda, H. (2008). Visualization by X-ray tomography of void growth and coalescence leading to fracture in model materials. *Acta Materialia*, 56(12):2919–2928.

Yan, X. H. and Leahy, R. M. (1991). Derivation and analysis of a filtered backprojection algorithm for cone beam projection data. *IEEE transactions on medical imaging*, 10(3):462–72.

Youssef, S., Maire, E., and Gaertner, R. (2005). Finite element modelling of the actual structure of cellular materials determined by X-ray tomography. *Acta Materialia*, 53(3):719–730.

Zhang, J. (2003). *Development of a High Resolution 3D Computed Tomography System*. Master of science, Iowa State University.